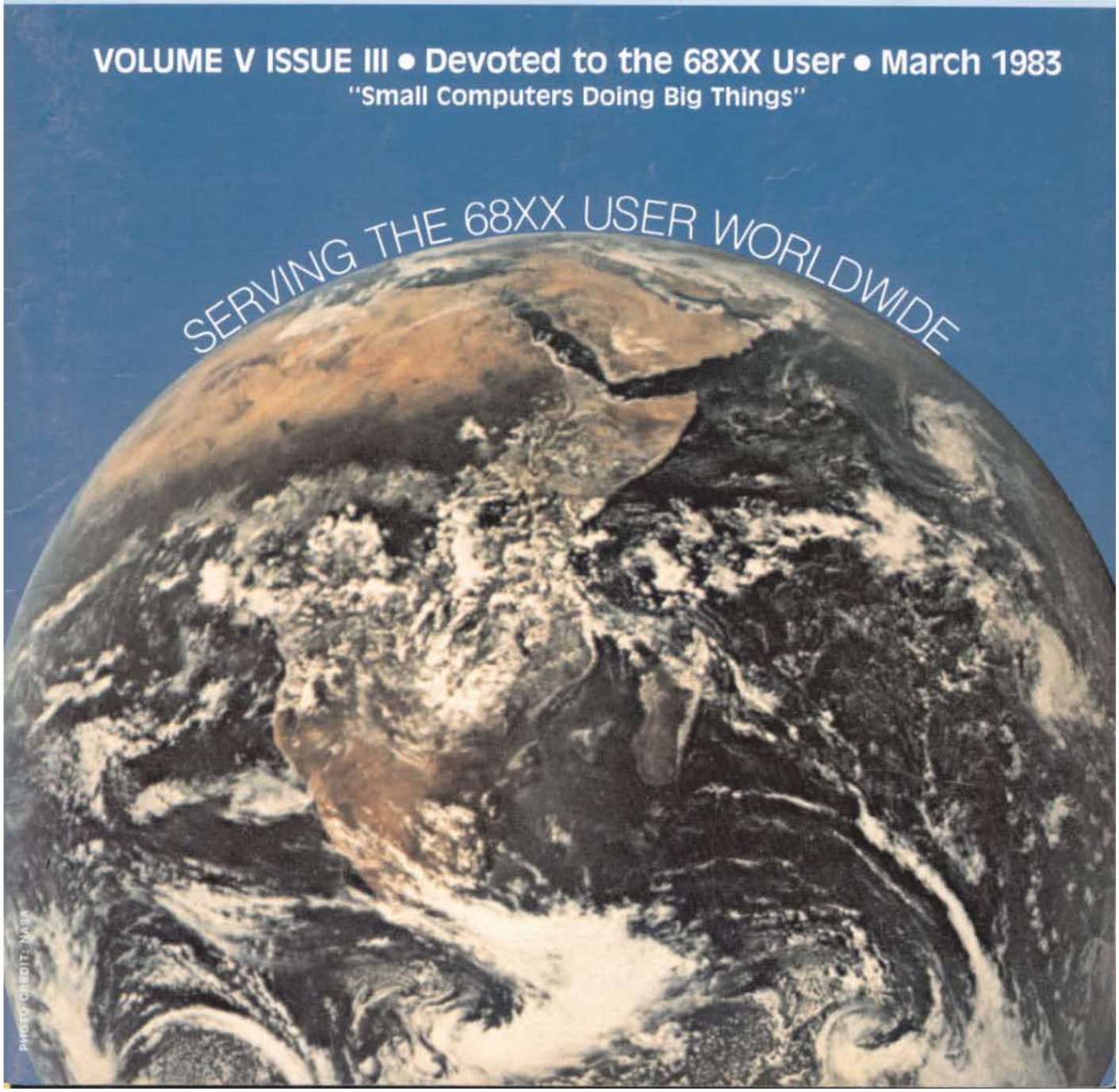
'68' MICRO JOURNAL

$2.95US

Australia	A $ 4.75	New Zealand	NZ $
Singapore	S $ 9.45	Hong Kong	H $
Malaysia	M $ 9.45	Sweden	30

VOLUME V ISSUE III • Devoted to the 68XX User • March 1983

"Small Computers Doing Big Things"



YOUR CHOICE-smart either way

- Over 140 software driven functions
- 82 x 24 or 82 x 20 screen format — software selectable
- High resolution 7 x 12 matrix characters — P-31 green phosphor
- Upper/lower case character set — plus graphics character set
- 56-key alphanumeric keyboard — plus 12-key cursor, numeric pad
- Internal editing functions — insert, delete, scroll, roll, slide, etc.
- Parallel printer I/O port
- 50 to 38,400 baud operation — programmable
- Cursor type, cursor position, print control characters, protected fields, shift inversion, dual intensity and many other features

8212 — twelve-inch diagnonal screen or **8209** — nine-inch diagnonal screen

SWTPC

SOUTHWEST TECHNICAL PRODUCTS CORPORATION
219 W. RHAPSODY
SAN ANTONIO, TEXAS 78216 (512) 344-0241

The UniFLEX™ Operating System extracts every last drop

from the 8 bit 6809 microprocessor allowing it to outperform many 16 bit systems

With the UniFLEX™ Operating System, the 8 bit 6809 microprocessor can perform as well as larger CPUs in a multi-user, multi-tasking environment.

Independently developed from the ground up, UniFLEX™ closely models the features found in the UNIX™ Operating System. And in two years of use, UniFLEX™ has proven the abilities of the 6809 to perform large system functions when incorporated into a properly designed mainframe.

Some of the features supported include:

- full multi-user, multi-tasking capabilities
- hierarchical file systems
- device independent I/O
- four Gigobyte disk capacities
- full file protection
- inter-task communication via pipes
- I/O redirection
- task swapping for efficient memory usage
- full random-access files
- comprehensive shell command language
- foreground-background jobs
- electronic mail and printer spooling
- system accounting facilities

The support software currently available for use under UniFLEX™ is extensive. A sampling of the programs available includes:

- native C compiler (full implementation)
- native Pascal compiler
- FORTRAN 77 ANSI Subset compiler
- COBOL compiler with ISAM files, Report Writer & Sort/Merge
- Extended BASIC interpreter
- Extended BASIC precompiler
- text editing and processing software
- enhanced printer spooler
- variety of absolute and relocatable assemblers
- debug and diagnostic packages

Technical Systems Consultants, Inc. also offers a line of single user FLEX™ software products for 6800 and 6809 processors. For those having an absolute need for a 16 bit processor, UniFLEX™ will be available through OEM licensing arrangements for the 68000 microprocessor. Please call or write for additional information on individual products or OEM licensing arrangements.

UNIX™ is a trademark of Bell Laboratories.
FLEX™ and UniFLEX™ are trademarks of Technical Systems Consultants, Inc.

technical systems consultants, inc.

111 Providence Road
Chapel Hill, North Carolina 27514
(919) 493-1451

'68'

MICRO JOURNAL

Portions of text prepared using the following.

SWTPC 6800-6809-DMAF2-CDS1-CT82-Sprint 3
Southwest Technical Products
219 W. Rhap ody
San Antonio, Texas 78216

EDITOR - WORD PROCESSOR
Technical Systems Consultants, Inc.
Box 2573, W. Lafayette, IN 47906
FLEX is TM of TSC

GIMIX Super Mainframe-Assorted memory boards
GIMIX Inc.
1337 We t 37th Place
Chicago, Il 60609

CONTENTS

Vol. V, Issue III MARCH'83

Send All Correspondence To:

Computer Publishing Center
68 MICRO JOURNAL
5900 Cassandra Smith
PO Box 849
Hixson, TN 37343
615 842-4600

68' Micro Journal is published 12 times a year by Computer Publishing Inc. Second Class Postage Paid ISSN 0194-5025 at Hixson, Tenn. and additional entries. Postmaster: send Form 3579 to 68' Micro Journal, PO Box 849, Hixson, Tennessee.

SUBSCRIPTION RATES

USA

1-Year $24.50 2-Years $42.50 3-Years $64.50

FOREIGN

See Page 52

Items Submitted for Publication

Articles submitted for publication should be accompanied by the authors **full name, address, date and telephone number.** It is preferred that articles be submitted on either 5 or 8 inch diskette in TSC Editor format or STYLO format. All diskettes will be returned.

The following TSC Text Processor commands **ONLY** should be used (due to our proportional processor): .sp space, .pp paragraph, .fi fill and .nf no fill. Also please do not format within the text with multiple spaces. The rest we will enter at time of editing.

STYLO commands are all acceptable except the .pg page command, we print edited text files in continous text.

All articles submitted on diskettes should be in TSC FLEX™ format, either FLEX2 6800, or FLEX9 6809 any version.

If articles are submitted on paper they should be on white 8X11 bond or better grade paper. No hand written articles (hand written or drawn art accepted). All paper submitted articles will be photo reproduced. This requires that they be typed or produced with a dark ribbon (no blue), single spaced and type font no smaller than 'elite' or 12 pitch. Typed text should be approximately 7 inches wide (will be reduced to column width of 3 1/2 inches). **Please u e a dark ribbon!**

All letters to the editor should also comply with the above and bear a signature. Letters of 'gripes' as well as 'praise' are solicited. We attempt to publish all letters to the editor verbatim, however, we reserve the right to reject any submission for lack of 'good taste'. We reserve the right to define what constitutes 'good teste',

Advertising: Commercial advertisers please contact the 68 Micro Journal advertising department for current rate sheet and requirements.

Classified: All classified must be non-commercial. Maximum 20 words per classified ad. Those consisting of more than 20 words should be figured at .35 cents per word. 20 words or less $7.50 minimum, one time, paid in advance. No classified ads accepted by telephone.

Intelligent Serial I/O Processor Board Now Available

The GIMIX Intelligent Three-port RS-232C Serial I/O Interface can significantly increase throughput of a multi-user system by reducing the number of interrupts between user terminals and the host CPU. The Intelligent I/O Board accomplishes this by buffering data transfers between system and users and preprocessing of the data.

Appropriate on-board software and operating system drivers are required. Software and drivers for OS-9 Level 2 will be available shortly from GIMIX.

- INDEPENDENT ON-BOARD **2MHZ 68B09** CPU
- UP TO **20K** OF ON-BOARD MEMORY (EPROM and RAM)
- BUFFERED DATA TRANSFER BETWEEN HOST AND ON-BOARD CPU USING A **Z8038** FIO I/O INTERFACE UNIT
- THREE **RS-232C** SERIAL I/O PORTS **(6551As)** WITH SOFTWARE SELECTABLE BAUD RATES, WORD LENGTH, STOP BITS, PARITY

Standard Version Including 4K RAM (Without Software) $438.11

— ***PARALLEL VERSION COMING SOON*** —

Uniflex For GIMIX Winchester Systems

TSC will be providing UniFLEX compatible with GIMIX Winchester systems. The NEW versions of UniFLEX for use with the Winchester systems will be delivered on 5" media as well as 8" media.

GIMIX 30 Pin Prototyping Board Now Available

- Double sided with plated thru holes and gridded power and ground lines.
- 8 rows of pads on .100 x .300 centers: up to 41 fourteen pin ICs.
- Accepts standard 6, 8, 14, 16, 20, 24, and 40 pin DIP devices.
- The entire top edge has pads for .100 x .100 header (ribbon) connectors.
- Pads for solder connections or .100 center headers on all 30 bus lines.
- Accepts 3 TO-220 regulators, 1 on the +8V & 1 ea. on the +/– 16V lines.
- Provisions for decoupling caps distributed throughout the array.
- Can be used with wire wrap, wiring pencil, solder wiring, etc.

With gold bus connectors and heat sinks (unassembled) $38.33

Now Available From GIMIX

(U.S. & Canada Only)

THE WINDRUSH EPROM PROGRAMMER

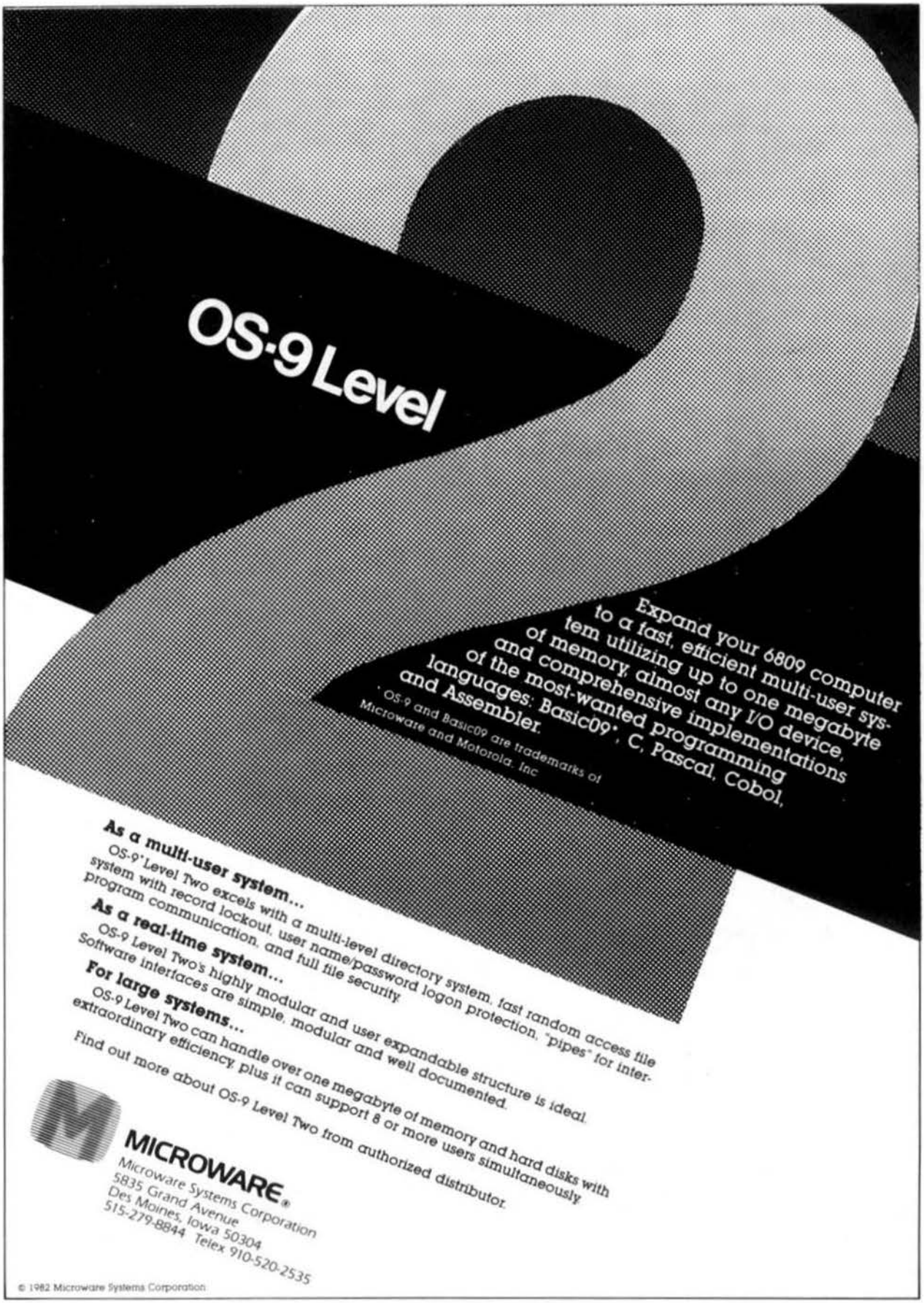
- ★ Probably the most versatile EPROM PROGRAMMER available. Interface & software for EXORcisor - II (fully addressable) and S-50 bus systems.
- ★ PROGRAMS AND VERIFIES 2508/2708, 2516/2716 (SINGLE AND TRI-VOLT TYPES) 2532, 2732, 2732A, 2564, 2764 and the 128K TMS2528 (16K x 8)
  - - - - - - - WITHOUT ADDITIONAL 'PERSONALITY' MODULES - - - - - - -.
- ★ PROGRAMMER extends out to your work area via 5' of twisted pair cable.
- ★ EXTENSIVE COMMANDS MENU...MOVE DATE, READ, PROGRAM, VERIFY EPROMS, EXAMINE/CHANGE BUFFER, FORMATTED DUMP OF BUFFER, FILL BUFFER.
- ★ Fully documented user's manual w/schematics & theory of operation. Professionally finished PCBs w/solder resist & component overlay.
- ★ SOFTWARE AVAILABLE FOR FLEX 2/9, SSB, OS-9 (LVL 1 NOW, LVL 2 LATER) and MDOS...All source files supplied. Specify disk size please!

NOTE: One version is supplied FREE. Extra versions: $25.00 each.

S-30 Interface/Programmer/Baseplate/Cable $375.00
EXORcisor Interface/Programmer/Baseplate/Cable $395.00

1337 WEST 37th PLACE
CHICAGO, ILLINOIS 60609
(312) 927-5510
TWX 910-221-4055

GIMIX INC.


OS-9 Level
Expand your 6809 computer to a fast, efficient multi-user system utilizing up to one megabyte of memory, almost any I/O device, and comprehensive implementations of the most-wanted programming languages: Basic09*, C, Pascal, Cobol, and Assembler.
* OS-9 and Basic09 are trademarks of Microware and Motorola, Inc.
As a multi-user system...
OS-9* Level Two excels with a multi-level directory system, fast random access file system with record lockout, user name/password logon protection, "pipes" for inter-program communication, and full file security.
As a real-time system...
OS-9 Level Two's highly modular and user expandable structure is ideal. Software interfaces are simple, modular and well documented.
For large systems...
OS-9 Level Two can handle over one megabyte of memory and hard disks with extraordinary efficiency, plus it can support 8 or more users simultaneously.
Find out more about OS-9 Level Two from authorized distributor.
MICROWARE®
Microware Systems Corporation
5835 Grand Avenue
Des Moines, Iowa 50304
515-279-8844 Telex 910-520-2535
© 1982 Microware Systems Corporation

NOW...FROM THE WORLDS LARGEST SUPPLIER OF SOFTWARE...

Yours For Only $3 Or FREE In Jan. 83 Color Computer News Magazine

COMES THE WORLDS LARGEST SOFTWARE CATALOG

SEE THE LATEST REVIEWS OF OUR SOFTWARE

November '82 **MICRO**
"FLEX and the TRS-80 Color Computer"
by Ronald Anderson, Page 23.

November '82 **80 MICRO**
"Color Forth"
by Jake Commander, Page 45.

November '82 **68 MICRO JOURNAL**
"CC FORTH"
by James Perotti, Page 19.

October '82 **RAINBOW**
A comparison of FHL Color Flex to 68 Micro Journal's (Data-Comp) FLEX, Page 64.

February '83 **80 MICRO**
Read the review of our DBASIC for FHL Color FLEX!

March '83 **80 MICRO**
FHL Color FLEX will be the feature review!!!

SEE OUR ADS IN

Color Computer News
(5 pages)
REMarkable Software
P.O. Box 1192
Muskegon, MI 49443
US $21.00 per year

The Rainbow
(5 pages)
5803 Timberridge Dr.
P.O. Box 209
Prospect, KY 40059
US $18.00 per year
US $22.00 Canada/Mexico
US $31.00 Foreign - surface mail
US $49.00 Foreign - airmail

80 Micro
(1 page)
80 Pine Street
Peterborough, NH 03458
US $25.00 per year
US $27.97 Canada/Mexico
US $44.97 Foreign

68 Micro Journal
(1 page)
5900 Cassandra Smith
P.O. Box 849
Hixson, TN 37343
US $24.50 per year
US $42.50 per 2 years
US $64.50 per 3 years

FHL FRANK HOGG LABORATORY

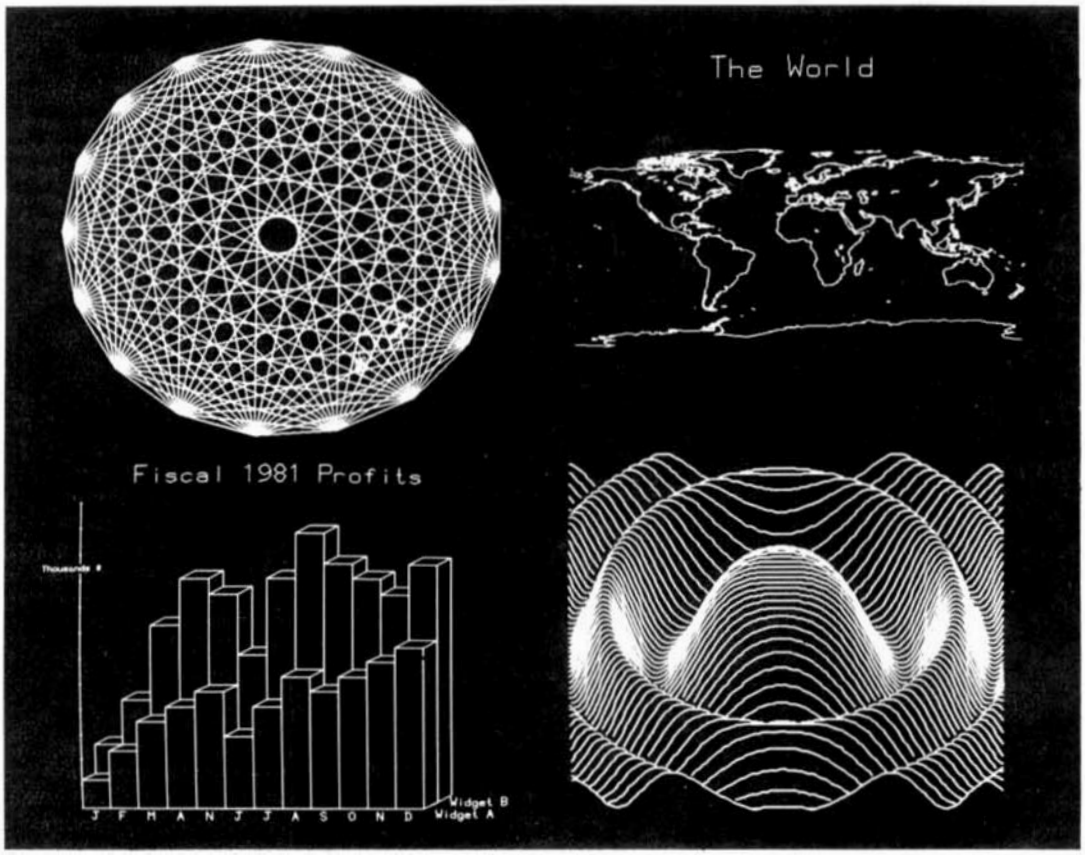
THE REGENCY TOWER • SUITE 215 • 770 JAMES ST. • SYRACUSE, NY 13203
PHONE (315)474-7856 • TELEX 646740

ANNOUNCING ElectroScreen™

only $595

the Superior Alternative

to the Traditional Alphanumeric Terminals



The ElectroScreen™ Intelligent Graphics Board Features:

Graphics

- 512 x 480 resolution bit-mapped display
- Interleaved memory access — fast, snow-free updates

Intelligence

- 6809 on-board mpu
- 6K on-board firmware
- STD syntax high level graphics command set
- Removes host graphics software burden
- Flexible text and graphics integration
- Multiple character sizes
- User programs can be run on-board

Terminal

- Terminal emulation on power-up
- 83 characters by 48 lines display
- Easy switching among user-defined character sets
- Fast hardware scrolling

Additional Features

- SS-50C and SS-64 compatible board
- Board communicates with host through parallel latches
- Composite and TTL level video output
- 8 channel 8 bit A/D converter
- Board occupies 4 address bytes

See your dealer today!

The ElectroScreen manual is available for $10. credited toward purchase of the board.

The ElectroScreen has a 90 day warranty from purchase date.

Dealers, please contact us for our special introductory package.

Privac Inc (703) 671-3900

3711 S. George Mason Dr., Falls Church, Va. 22041

Flex User Notes

Ronald W. Anderson
3540 Sturbridge Court
Ann Arbor, MI 48105

MORE TEMPERATURE APPLICATION

Last time, I presented some hardware for a system to monitor temperatures around a house, and provide information to the computer on actual and desired temperatures in each room. I promised some software this time. Here is the first installment of it. I've done the test software in both Pascal and Assembler. Both listings are presented here.

As a first step, these programs initialize the A/D card, and then fall into a repeat forever loop to read the 16 channels and present the readings on the CRT. This will be useful to test out the A/D converter and wiring of the hardware at the input side of the system. Note that the Pascal program is not all that much shorter than the Assembler program in terms of pages. I think you will agree that it is easier to follow.

Pascal Program

First we will look at the Pascal program. This one is in Lucidata Pascal. The implementation is important, because it is necessary to use some of the language's extensions to access the A/D directly. Both Lucidata and OmegaSoft Pascal implementations have sufficient extensions to allow such easy access to hardware ports. Lucidata has BYTE (8 bit) data, which is a subrange of integer. It is unsigned and has a range of values of 0 to 255 or 0 to $FF. Lucidata also has a HEX data type and allows assigning Hexadecimal values by preceding the value with a dollar sign, using the same symbol as the standard Motorola assembler.

First, I have declared some constants that equate A/D channel numbers with rooms and functions. Next is the conversion factor for a span of 36 degrees and an 8 bit A/D with its range of 0 to 255. The minimum temperature of the range is specified as MINTEMP, and the Hex constants that are used to turn the START convert pulse on and off, are equated. Some constants to allow clearing the screen and homing the cursor for my ADM-3A are also defined.

Lucidata uses what are called "Pragmats" to instruct the complier, just as Pseudo Operations are used in an Assembler. (*$A = $E00C *) assigns the next variables at consecutive addresses starting at $E00C, which is where my A/D board is located. When we get to the Assembler version, you will note a great deal of similarity. I've done that purposely so you might see the correspondence between the two programs.

After declaring the variables for the A/D (note that they are declared as BYTE variables), there is one INTEGER variable that is just used as a counter in a delay loop.

The procedure INITIALIZE simply sets up the PIA on the A/D board. The setup is as described in the JPC manual. The function CONVERT reads the channel whose number is passed to it, and returns the temperature in degrees and tenths, doing the conversion calculation in its last line.

The main program simply clears the screen, reads each channel and outputs the temperature to the screen, with identifying labels (words such as Living Room), homes the cursor and repeats indefinitely.

Assembler Program

If you are very familiar with Assembler, this one might help you understand the Pascal version (and vice versa). Since I plan on running the tests of the A/D and thermistor circuits in my system, I used FLEX PUTCHR to output to the CRT. The four addresses in the PIA for the A/D are equated using the same names as in the Pascal Program. Channel numbers are assigned in the same way, also.

At that point, the Assembler program is occupied with doing a function that is built in to Pascal. The first actual program bytes are a table of BCD values that are used in the BINARY to BCD to ASCII conversion routine. The convert pulse on and off constants are defined, and unity gain control for the A/D.

Next is the CONVRT subroutine that drives the A/D. It is passed the channel number in ACCA and it returns the A/D output in ACCA. There is a delay loop subroutine, and then the INIT subroutine. The CONVRT and INIT subroutines do exactly the same thing as the Pascal versions, except that the data is not scaled in CONVRT, but passed back to the main program for scaling.

The Assembler version of course runs faster than the Pascal version, and I found that I could clear the screen and rewrite it quite fast. CLRSCR is the subroutine that does the clear function.

PCRLF is an internal CR/LF generator that avoids getting tangled with the FLEX Pause feature. BDCON converts a 10 bit binary number to decimal and then to ASCII code for output to the CRT. The number is passed to it in the D register and results are returned on the user stack. The idea is to add the decimal value from the table for each bit in the binary number that is on. Of course the DAA instruction is used frequently.

The main program initializes the A/D and then enters its repeat forever loop. It clears the screen, sets up a counter for the channel conversions, reads and scales the data. The correct scaling factor is actually 360/255. I've made a slight approximation and used 360/256, which reduces to 45/32. The result is an error of 0.2 degrees at maximum. That is, the readings vary from 50.0 to 85.8 rather than 86 degrees F. The minimum temperature is added to the scaled value from the A/D (ADDD #500) and the result is converted to ASCII and output with a decimal point inserted before the low order digit. After all readings are output, there is a delay loop that keeps the display from flickering and changing too quickly. There is no way for the loop to terminate except that it checks the terminal serial interface receive buffer for an escape. If it finds one there, it exits to FLEX Warmstart address. If you decide that my Assembler programming has been influenced by Pascal, you are probably correct. I think the influence is for the better.

Next month we will look at the output side of the system. We will store temperatures and set commands in an array and compare them for each room. The differences will cause control outputs to go on and off to open and close baffles in the heat ducts or solenoid valves in hot water heating pipes.

It is important to use comparisons that will allow a furnace to cycle not too frequently. The temperature differential from heat on to heat off for each room should be 2 degrees or so. I hope this subject will be a good introduction to using a computer to control hardware. By now, you should be seeing some of the possibilities of such a system. The programmer can "fine tune" such things as on-off differentials to make the system run well. It may be necessary to wait for at last two areas to call for heat before starting the furnace. Also we will have to consider what to do when the demand for heat ends but the furnace blower still wants to

circulate warm air. We can't just shut all the baffles and prevent the furnace from cooling itself. That would be both dangerous and inefficient. Most likely we will leave the last baffles open until the blower shuts off. That will, of course, mean that we will have to input information from the furnace control.

Just one other thing I should say, is that you don't have to actually build this system to benefit by the discussion here. It is highly unlikely that even I will build it. (If I can find some very inexpensive actuators for the baffles, I might.) The point is that one reader wrote and asked me to give an example of doing a control function with a 6809 system. I can't very well use programs from industrial controls and instruments as examples for a couple of reasons. First of all, some of the theory behind them is rather deep, and it would be very hard to explain. Secondly, the companies that I work for regard their electronics as trade secrets. They feel that some of these designs give them an edge over what the competition has to offer, and would object strenuously to having any details published for their competitors to see.

Did I have any problems writing these two programs? You betcha. I did the assembler version first, and managed to get it running, starting from scratch in about 2 hours. I noted that somehow the screen displayed the last channel up top on the first line. It took three more hours to figure out that I had initialized the A/D wrong and I was reading the value from the previous conversion rather than the current one. The Pascal program went a bit faster, but I had a space in the Pragmat line that assigned absolute addressing mode, so that the command was ignored. The A/D registers were assigned on the data stack, and it took me an hour or more to find that little peculiarity. Of course I had the advantage of having written programs to use this particular A/D converter card previously.

80 Track Drive Compatibility Problems

Have you tangled with an 80 track drive or a disk made on one yet? Seemingly strange things happen when you try to read disks made on an 80 track drive, even if the disk is formatted in the "double stepping" mode with 40 tracks. What's the problem? The main one is one that I missed altogether. Suppose a disk has been with 80 tracks on an 80 track drive or with 40 on a normal 40 track drive. Now someone puts it on an 80 track drive and formats it using double stepping. If it had previously been formatted on an 80 track drive, there are 40 tracks of information between (or along side of) the 40 newly formatted tracks. The 80 track drive doesn't care, but if you try to read that disk on a 40 track drive, you will read two tracks at a time! Same problem if you have previously formatted the disk on a 40 track drive. The 80 tracker will only overwrite half of each track as it formats the disk. A 40 track drive will read both the old and the new tracks.

If that isn't bad enough, suppose you take a new disk that has been bulk erased by the manufacturer (most come that way). Now format it double stepping an 80 track drive. If you have a good 40 track drive you might be able to read it fine. However, the tracks are narrower, and will most likely produce less signal in the head of a 40 track drive than would the signal from a normal 40 track formatted disk.

So now we have 35 track, 40 track, 80 track, single sided, double sided, single density, and double density formats all on 5 inch disks. We have IBM standard format, (GIMIX adheres to that standard) SWTPC FLEX format (almost but not quite IBM standard), Miniflex and CP/M format (128 bytes per sector compared to IBM's 256). Anyone out there have any ideas how even we FLEX users can come up with a standard disk format for exchange purposes? I would think the software suppliers would go nuts trying to keep up with all the possibilities. I guess we can all write and read 35 track single sided single density disks, if we have a version of FLEX running. That will have to be the default standard for some time, I suppose.

```
1.00= NAM THERM
2.00= TTL THERMISTOR THERMOSTAT
3.00= OPT PAG
4.00= PAG
5.00=*
6.00=*
7.00=* PROGRAM TO DRIVE JPC 16 CHANNEL A/D CARD TO READ
8.00=* THERMISTOR TEMPERATURE SENSORS AND POTENTIOMETER
9.00=* TEMPERATURE SETTINGS. AND CONTROL BAFFLES IN HEAT
10.00=* DUCTS.
11.00=*
12.00=*
13.00=* SYSTEM EQUATES
14.00=*
15.00=* TERMINAL I/O FOR TESTING OF TEMPERATURE MEASUREMENT AND
16.00=* POTENTIOMETER READING ROUTINES
17.00=*
18.00=PUTCHR EQU $CD18
19.00=WARMS EQU $CD03 FLEX WARMSTART
20.00=ATOD EQU $E00C SET THIS TO MATCH YOUR PORT ADDRESS FOR A/D
21.00=STATUS EQU ATOD+1
22.00=SETUP EQU ATOD+2
23.00=CONTRL EQU ATOD+3
24.00=*
25.00=* "READABILITY EQUATES"
26.00=*
27.00=* CHANNEL NUMBERS FOR A/D
28.00=*
29.00=LR_TMP EQU 0 LIVING ROOM
30.00=LR_SET EQU 1
31.00=DR_TMP EQU 2 DINING ROOM
32.00=DR_SET EQU 3
33.00=FR_TMP EQU 4 FAMILY ROOM
34.00=FR_SET EQU 5
35.00=KT_TMP EQU 6 KITCHEN
36.00=KT_SET EQU 7
37.00=MB_TMP EQU 8 MASTER BEDROOM
38.00=MB_SET EQU 9
39.00=B2_TMP EQU 10 SECOND BEDRODM
40.00=B2_SET EQU 11
41.00=B3_TMP EQU 12 THIRD BEDROOM
42.00=B3_SET EQU 13
43.00=WK_TMP EQU 14 WORKSHOP
44.00=WK_SET EQU 15
45.00=LAST EQU 15 LAST USED CHANNEL
46.00=*
47.00=*
48.00=CONVT EQU $3C SET HANDSHAKE HIGH
49.00=CONOFF EQU $34 SET HANDSHAKE LOW
50.00=GAIN EQU $10 FOR UNITY GAIN
51.00=*
52.00=* START OF PROGRAM
53.00=*
54.00=START LBRA BEGIN
55.00=*
56.00=* DECIMAL VALUE TABLE FOR BINARY TO DECIMAL CONVERSION
```

```
 57.00=*
 58.00=BCDTBL EQU *
 59.00= FDB $0001
 60.00= FDB $0002
 61.00= FDB $0004
 62.00= FDB $0008
 63.00= FDB $0016
 64.00= FDB $0032
 65.00= FDB $0064
 66.00= FDB $0128
 67.00= FDB $0256
 68.00= FDB $0512
 69.00=*
 70.00=* CONVERT SUBROUTINE
 71.00=*
 72.00=* ENTER WITH CHANNEL NUMBER IN ACCA
 73.00=* RETURNS DATA IN ACCA
 74.00=*
 75.00=CONVRT ORA #GAIN
 76.00= STA SETUP
 77.00= LDA #CONVT
 78.00= STA CONTRL ADDRESS LATCHED ON HIGH
 79.00= LDA #CONOFF
 80.00= STA CONTRL STARTS CONVERT
 81.00=CONVT2 LDA STATUS
 82.00= BPL CONVT2 WAIT FOR CONVERSION DONE
 83.00= LDA ATOD GET DATA
 84.00= RTS
 85.00=*
 86.00=*
 87.00=* INITIALIZE PIA ON A/D
 88.00=*
 89.00=INIT CLR STATUS
 90.00= CLR CONTRL
 91.00= CLR ATOD SET FOR INPUTS
 92.00= LDA #6
 93.00= STA STATUS TO ACCESS DATA REGISTER
 94.00= LDA #$FF
 95.00= STA SETUP OUTPUT FOR MUX CHANNEL
 96.00= LDA #CONOFF SET UP HANDSHAKE
 97.00= STA CONTRL
 98.00= LDU #$BFFF SET USER STACK TO HIGH RAM
 99.00= RTS
100.00=*
101.00=* CLEAR SCREEN SUBROUTINE
102.00=* TERMINAL DEPENDENT.  SHOULD CLEAR SCREEN AND HOME CURSOR
103.00=* THIS ONE IS FOR ADM-3A
104.00=*
105.00=CLRSCR LDA #$1A
106.00= JSR PUTCHR
107.00= LDA #0
108.00= JSR PUTCHR
109.00= LDA #0
110.00= JSR PUTCHR  ADM NEEDS TIME TO CLEAR SCREEN
111.00= LDA #0
112.00= JSR PUTCHR  3 NULLS AT 19.2K BAUD
113.00= RTS
114.00=*
115.00=* CR/LF ROUTINE
116.00=*
117.00=PCRLF LDA #$0D
118.00= JSR PUTCHR
119.00= LDA #$0A
120.00= JSR PUTCHR
121.00= RTS   ADD NULLS IF REQUIRED BY YOUR OUTPUT DEVICE
122.00=*
123.00=*
124.00=* BINARY TO BCD TO ASCII CONVERSION
125.00=* BINARY PASSED IN ACCD
126.00=* RESULT PUSHED ON USER STACK. HIGH ORDER LAST
127.00=*
128.00=BDCON LEAX BCDTBL,PCR
129.00= PSHU D
130.00= CLRA
131.00= CLRB
132.00= PSHU D SPACE FOR RESULT
133.00= LDB #10 LOOP COUNT
134.00=BDC1 LSR 2,U ROTATE BINARY NUMBER RIGHT
135.00= ROR 3,U
136.00= BCC BDC2 SKIPP ADD IF BINARY BIT WAS ZERO
137.00= LDA 1,U ADD DECIMAL VALUE IF BINARY WAS 1
138.00= ADDA 1,X LOW ORDER BYTE
139.00= DAA
140.00= STA 1,U
141.00= LDA 0,U
142.00= ADCA 0,X
143.00= DAA
144.00= STA 0,U
145.00=BDC2 LEAX 2,X POINT AT NEXT DECIMAL VALUE
146.00= DECB LOOP COUNTER
147.00= BNE BDC1 DONE YET?
148.00= PULU D IF DONE GET RESULT
149.00= LEAU 2,U REMOVE INPUT VALUE
150.00= PSHS B SAVE LOW ORDER BYTE
151.00= ANDB #$0F MASK SECOND BCD DIGIT
152.00= ADDB #$30 MAKE IT ASCII
153.00= PSHU B
154.00= PULS B
155.00= LSRB
156.00= LSRB
157.00= LSRB
158.00= LSRB
159.00= ADDB #$30 SECOND DIGIT
160.00= PSHU B
161.00= ADDA #$30 HI ORDER DIGIT
162.00= PSHU A
163.00= RTS
164.00=*
165.00=* MAIN PROGRAM
166.00=*
167.00=* INTERVAL 0 TO 255 ON A/D WILL BE 50 TO 86 F.
168.00=* A SPAN OF 36 DEGREES.
169.00=* WE WILL MULTIPLY A/D BY 360/256 TO GET DEGREES
170.00=* AND TENTHS.  360/256  REDUCES TO 45/32
171.00=*
172.00=*
173.00=BEGIN BSR INIT
174.00=MEASUR LBSR CLRSCR
175.00= CLRB
176.00=MEASUR PSHU B COUNTER FOR CONVERSIONS
```

```
177.00= LDA 0,U
178.00= LBSR CONVRT
179.00= LDB #45
180.00= MUL RESULT IN ACCD
181.00= ASRA
182.00= RORB
183.00= ASRA
184.00= RORB
185.00= ASRA
186.00= RORB
187.00= ASRA
188.00= RORB
189.00= ASRA
190.00= RORB NOW WE'VE DIVIDED BY 32 AND HAVE TEN BITS OF INFO
191.00= ADDD #500 (50.0)  ADD IN THE LOW TEMP LIMIT
192.00= LBSR BDCON  VALUE IN ACCD FOR CONVERSION
193.00= PULU A
194.00= JSR PUTCHR
195.00= PULU A
196.00= JSR PUTCHR
197.00= LDA #'. DECIMAL POINT
198.00= JSR PUTCHR
199.00= PULU A LAST DIGIT
200.00= JSR PUTCHR
201.00= LDA #$20 SPACE
202.00= LBSR PUTCHR
203.00= PULU B COUNTER
204.00= BITB #1 SEE IF CHANNEL IS ODD
205.00= BEQ MEASU2 NO CR. PUT TEMP AND SET ON A LINE
206.00= JSR PCRLF
207.00=MEASU2 INCB
208.00= CMPB #LAST
209.00= LBLE MEASUR
210.00= LDX #$FFFF
211.00=WAIT LEAX -1,X
212.00= BNE WAIT
213.00= LDA $E005 KEYBOARD SERIAL PORT RECEIVE BUFFER
214.00= CMPA #$1B ESCAPE
215.00= LBNE MEASU1
216.00= JMP WARMS
217.00= END START TRANSFER ADDRESS
218.00=*
219.00=* THIS IS A RUN FOREVER PROGRAM
220.00=*
```

```
1.00=PROGRAM TEMPMEAS (INPUT, OUTPUT);
2.00=
3.00=CONST
4.00=
5.00=( CHARACTERS FOR SCREEN CLEAR AND HOME STRINGS )
6.00=
7.00=   CLEAR = CHR(26);
8.00=   HOME  = CHR(30);
9.00=   NULL  = CHR(0);
10.00=
11.00=( CHANNEL NUMBERS FOR A/D CONVERTER PHYSICAL CONNECTIONS )
12.00=
13.00=   LR_TEMP    = 0;
14.00=   LR_SET     = 1;
15.00=   DR_TEMP    = 2;
16.00=   DR_SET     = 3;
17.00=   FR_TEMP    = 4;
18.00=   FR_SET     = 5;
19.00=   RT_TEMP    = 6;
20.00=   RT_SET     = 7;
21.00=   MB_TEMP    = 8;
22.00=   MB_SET     = 9;
23.00=   B2_TEMP    = 10;
24.00=   B2_SET     = 11;
25.00=   B3_TEMP    = 12;
26.00=   B3_SET     = 13;
27.00=   WK_TEMP    = 14;
28.00=   WK_SET     = 15;
29.00=
30.00=   DEGREES = 0.141177; ( CONVERTS 255 SCALE TO 36.0  SCALE )
31.00=   MINTEMP = 50.0;
32.00=   CONVT = $3C;
33.00=   CONOFF = $34;
34.00=
35.00=VAR
36.00=
37.00=   (*$A= $E00C *) ( ASSIGN VARIABLES TO A/D PORT ADDRESSES )
38.00=
39.00=   ATOD,
40.00=   STATUS,
41.00=   SETUP,
42.00=   CONTROL : BYTE ;
43.00=
44.00=   (*$S *) ( VARIABLES BACK TO STACK )
45.00=
46.00=   N : INTEGER;
47.00=
48.00=
49.00=
50.00=PROCEDURE INITIALIZE;
51.00=
52.00=BEGIN
53.00=   STATUS := 0;
54.00=   CONTROL := 0;
55.00=   ATOD   := 0;
56.00=   STATUS := 6;
57.00=   SETUP  := 255;
58.00=   CONTROL := CONOFF;
59.00=END;
60.00=
61.00=
62.00=
63.00=
64.00=FUNCTION CONVERT (CHANNEL : BYTE) : REAL;
65.00=
66.00=VAR
67.00=   N : INTEGER;
68.00=
69.00=BEGIN
70.00=   SETUP := CHANNEL;
71.00=   CONTROL := CONVT;
72.00=   FOR N := 1 TO 10 DO; ( DELAY LOOP )
73.00=   CONTROL := CONOFF;
74.00=   REPEAT UNTIL STATUS > 127;
75.00=   CONVERT := ATOD * DEGREES + MINTEMP;
76.00=END;
77.00=
```

```
78.00=
79.00=
80.00=BEGIN { MAIN PROGRAM }
81.00=
82.00=   INITIALIZE;
83.00=   WRITE (CLEAR,NULL,NULL,NULL);
84.00=
85.00=   REPEAT
86.00=
87.00=      WRITE (HOME,NULL,NULL,NULL);
88.00=
89.00=   WRITELN ('LIVING ROOM TEMP',CONVERT(LR_TEMP)  : 5:1,
90.00=           '   SET POINT',CONVERT(LR_SET)  : 5:1);
91.00=   WRITELN ('DINING ROOM TEMP',CONVERT(DR_TEMP)  : 5:1,
92.00=           '   SET POINT',CONVERT(DR_SET)  : 5:1);
93.00=   WRITELN ('FAMILY ROOM TEMP',CONVERT(FR_TEMP)  : 5:1,
94.00=           '   SET POINT',CONVERT(FR_SET)  : 5:1);
95.00=   WRITELN ('KITCHEN TEMP    ',CONVERT(KT_TEMP)  : 5:1,
96.00=           '   SET POINT',CONVERT(KT_SET)  : 5:1);
97.00=   WRITELN ('MASTER B.R. TEMP',CONVERT(MB_TEMP)  : 5:1,
98.00=           '   SET POINT',CONVERT(MB_SET)  : 5:1);
99.00=   WRITELN ('SECOND B.R. TEMP',CONVERT(B2_TEMP)  : 5:1,
100.00=          '   SET POINT',CONVERT(B2_SET)  : 5:1);
101.00=  WRITELN ('THIRD B.R. TEMP ',CONVERT(B3_TEMP)  : 5:1,
102.00=          '   SET POINT',CONVERT(B3_SET)  : 5:1);
103.00=  WRITELN ('WORKSHOP TEMP   ',CONVERT(WK_TEMP)  : 5:1,
104.00=          '   SET POINT',CONVERT(WK_SET)  : 5:1);
105.00=  UNTIL FALSE;
106.00=END.
```

RUMORS & SUCH

Note: These 'rumors' are exactly what we claim them to be, **RUMORS.** We feel that in most all instances, our sources are reliable. Over the past 5 years our 'rumors' have been far more accurate, than most published in other computer magazines. Therefore, I would trust that you, our readers, will accept them as such, and understand that although we 'know' most are accurate, we still classify them as such, despite the many hours spent verifying, to some degree or another, the veracity of our 'rumors'. Also I want to thank all of you who have, from time to time, passed on information, we all consider interesting.

RUMORS & SUCH

Well after being somewhat 'off-line' for about 2 months I finally am back to full bore, which is to say, here are some more rumors and other such trivia.

For the past 2 months I have been sorta low key around the office. Seems that at my age my bones can't take it as well as they used to. If you've never drug around with about 40 pounds of plaster of paris draped around your leg, well, you won't know what I mean. Actually I got a chance to catch up on my reading and motorcycle rebuilding (rebuilt a Harley Sportster to like new). Seems I really dazzled my grandson Chris, when I demonstrated to him the unapproved way to 'toss away' a motorcycle. The dumb part is that I was only moving about 4 or 5 MPH, when I decided to stop on a bed of wet leaves. Talk about surprise! What's even worse is that at my age and after riding the blooming things for over 35 years, I should have known better. Always has to be a first time.

What has this got to do with computers, you ask? Well it really doesn't, except I actually fall into a 'cast' (no pun intended), or average grouping, of our readers.

Last year I conducted an extensive survey for a large electronics manufacturer. They wanted answers to some of the silliest questions you ever heard, and a lot of very serious ones. Our survey should have produced a 'fudge factor' of less than 2% error, don't ask me why but that is what I am told by the Madison Avenue types who made up the questions. We surveye 1,602 readers, so based on a probable readership of some 25,000 (they figure about 2.1 readers for each magazine sold) that was **their** number of potential accuracy.

In addition to all the questions they asked, I added a few of my own (sure saves on the telephone bill), the results verified some suspicions I have had about the makeup of our readership. While I won't bore you with a lot of seemingly useless details, I thought you might be interested in what wo are, as concerns our other activities.

For the most part we still are a major percentage of 'do-it-yourself' types. No surprise when you consider where we came from. Over 60% of those surveyed are making, or are trying to make, a buck with computers, actually only about 40% were hobby only. Over 91% of us primarily with 68XX(X) systems. The average age is 37 and 84% married. The median income was $28,452.07 per year. Aside from a couple of doctors and other above average income garners, the majority derived their income (better than 51% of income) from their work experience with computers, electronics or allied pursuits. Less than 30% are into really big stuff, 370s, etc. None of these numbers hold up for the Color Computer, they are younger and broker, and work at a lot of other stuff.

We even asked about hobbies, and that is where more suspicions were confirmed. Actually I had a pretty accurate 'gut' feeling about our group, after 7 or 8 years spent talking to thousands of you, I should. It seems that there is a certain 'need to know' attitude among us. A large percentage have very allied attitudes about their hobby (actual or wished for). Of course, computers were at the head of the list. Also amateur radio, motorcycles and airplanes were up there. So I rediscovered that I fit the average. Being the honcho here at 68 MICRO JOURNAL and having access to about any 68XX computer I want, certainly has a big advantage. But I also 'live and breath' motorcycles, amateur radio and airplanes. Evidenced by the fact that I own 4 motorcycles and a 'nearly-like-new' 1962 Cessna 172 airplane. You would be surprised at the number of you who also own or operate motorcycles and airplanes. Also high on the list was computer applications for amature radio (not CB), building and flying model airplanes, skiing, hunting and fishing, jogging and bowling (a few jokers included 'girls' and one, 'boys'). The motorcycles, airplanes and amature radio (my call W4MQN) put me right in there, as it probably does a lot of you.

Some New 68XX(X) Products

Funny thing but it seems that the 68XX series of computer devices are finding their way into more advanced and complicated applications, than any other type of CPUs. It is no deep secret that we are not top dog in number of computers using microprocessor devices. It seems that 'we' are just recently being discovered. Which goes to show that 'we' knew all along what others are now finding out. Three new or recent entries are the new Heathkit robot, their robot and 6809 training course. The robot, named HERO 1, uses the 6808, which is a 6802 without RAM. The robot course is centered around the 6808 and of course their robot kit. The 6809 Course may foretell some new Heath products using that fine device.

Will have a lot more to say about these at a later date, but having put together one of their first CRT terminal kits (H9), I have some reservations. When they first announced the H9 CRT terminal I ordered the very first from their Atlanta outlet. Needed it for a special customer application and the kit seemed an easy way to get a terminal up and running with sufficient documentation to make necessary modifications. Well, when it arrived, less the CRT (took six weeks or so to get it, as I remember) we jumped into building the thing.

Right off the bat I began to suspicion that we were in trouble, especially when I discovered that we had to 'stick on' the characters to the top of each key. The board that came already assembled (to prevent construction errors) never did function properly. The main cable which came pre-assembled had over 5 wires in the wrong pins. Also to compound the fun the schematic was also wrong, but different wires! Spent a lot of time on the telephone with their engineers listening to how everything was o.k. and that I must have done something wrong. Four or five months later I got my hands on the 'revised' schematic and found the changes made, boy, those guys sure hated to 'fes up! Anyway, finally got the thing running, but at anything above 1200 baud it

scrolled with a jitter that would put a three legged horse to shame. The case was some kind of pressed plastic and the finish was painted on. It rubbed off in no time flat. The horz. sweep circuit ate transistors like candy, until we received the 'fix' (3 months later). All in all the end result was something less than satisfying. Actually Heath came up with 'fixes' for most of the worst 'bugs', but some things never did get straightened out.

In all fairness I must tell you that I have assembled other Heath kits. Most all worked as advertised and the documentation was excellent, in fact the **very best.** The H9 was actually the only Heath kit that really was a dud, in my experience (reckon everybody's entitled to one). I guess the lesson to be learned is **aware of early production** kits. So until I have more solid information concerning Heath's new 68XX products will hold any additional remarks. owever, we will be testing them, in our labs, so keep tuned in.

If they measure up to many of the other Heath products, then they should find ready acceptance among many of you. Robots are a coming thing and that Heath saw fit to use a 6808 for so complicated an application is no wonder to us. As to the course on the 6809, it just may be what a lot of you are looking for. So if you plan to try one of these, especially the robot, you might want to wait for our evaluation (heap of money). Anyway, I promise, to let you know.

Last year I mentioned that IBM had contracted for a large quantity of 68000 CPUs. Seems that they are attempting (or may already have, who knows) to build a 68000 370 type. High speed 68000s (16 Mhz or about) that are microcoded. Well, they havn't let much out about that project (also watch national Cash Register) but they have come out with a 68000 computer.

Actually it is **not** a true business machine, more a lab type computer. Heavy number cruncher, etc. However, just a little, here and there, and presto a full blown 68000 mainframe, ala IBM. It employs modular construction and comes with a 12" B&W CRT and 57 key keypad for about $5000-6000. When the printer/plotter, typewriter type keyboard and some other options are added, the price creeps up into the $10,000+ range. The modular units stack on top of one another (vertically) for minimum space. It can address up to 5 megabytes of memory.

A very important design aspect is the type of backplane or bus used. In this case IBM has adopted the Motorola Versabus™, which will probably do more to promote the Versabus, than any other thing that could occur to it at this point in time. In it's basic configuration it can hold 5 Versabus cards. Standard RAM is 128K with provisions for 128K of ROM. In addition to standard RS232 serial ports (3) and one 8 bit parallel port it provides for IEEE-488 interfacing.

Not much applications software or many languages presently available, but I imagine it won't be long. The entry of IBM into the 68XX(X) marketplace certainly will influence many existing 68XX(X) manufacturers, as to their R&D efforts.

IBM may be a hard act to follow, but if the future follows past track records, many smaller companies can compete well. The very size of IBM makes their market entry with almost any product slow, excepting their 'PC'. In the past smaller manufacturers have 'hit the market' far enough in advance to compete with IBM. Also as technology advances, the smaller firms can develope and market faster, than a larger outfit. All in all it forebodes good things for the 68XX(X) marketplace. However, for those existing 68XX(X) manufactures, it may revise their approach to the 68000 market. Especially concerning the Motorola Versabus. Wonder who will be the first with a S50 to Versabus interface? When it does evolve into a 'full blown' business applications machine, look for UNIX™ or a UNIX type operating system to soon follow, one we may recognize. 68000 interest seems to be exploding!

Also recently Apple demonstrated (for a few select dealers) their 'Lisa' 68000 computer. This one will have to make it on the Apple name, as the machine appears to leave a lot to be desired. It's big thing supposed to be 'mouse', hand held, that points to pictures and words, on the screen. No typing of commands, just position the cursor 'mouse' and execute a command. Sounds like a ten grand Mickey Mouse to me.

First off it is rumored that for a dealer to stock Lisa, that dealer must purchase or 'invest' close to $100,000.00, for inventory. Secondly, it is rumored to **only be able to address 1 Megabyte!!** The winchester capacity is rather small, by today's standards. The basic cost is somewhere around $10,000.00. All in all it appears that Apple is attempting to carry their 'home/hobby' attitudes (engineering and distribution) over to the real world of business computers. My guess is that the 'Big Blue' will teach Apple a thing or two.

A lot of Apple dealers (who won't or can't afford to stock Lisa) are reported to be 'up in arms' over the Apple Attitude. Oh well, apples are for 'eating' anyway, and 'Big Blue' and a lot of other 68000 manufacturers should do the munching. Maybe they (Apple and friends) can get their tax relief bill through Congress so they can give them away to schools, and what not.

During the past month we also received, for evaluation, the Epson HX-20, portable computer. Which will probably be sold by some of their printer dealers, and a lot of discount retailers. It uses a CMOS version of the 6802. 16K of RAM and 24K of ROM come standard, includes BASIC, which it powers up in. By 'POKEing' around I found it has a very interesting command table.

From the looks of the unit, it is a nice 'small' machine, despite it's rather high selling price ($795). A quality keyboard makes it a 'step up' over most pocket computers. Somewhat larger than most pocket computers, it will not fit in a shirt pocket, at least no shirt I own. However, it is battery backed-up and comes complete with printer. The display is a 4 line liquid crystal device. Beyond that there is not much more I can tell you at the present time. They sent no documentation with the unit initially. A few days ago we received a very preliminary copy of the users manual, after 5 unanswered telephone calls and a letter or two. **No** documentation on the ROM BASIC (Microsoft) or machine language utilization, or the several odd accessory plugs.

It may be just the thing some of you are looking for, but without any solid documentation, I can't tell you much more. When, and if, we receive the rest of the books that should go with this unit, will let you know. Until then your guess is as good as mine.

It is rumored that Texas Instruments will soon be using the 68000 in a new TI computer. Supposedly a dual 68000 machine that will be a 'high end' multi-user system. The schedule appears to be a 'hurry up' schedule, maybe to gain market access ahead of IBM's PC II and DEC's Professional 350. Seems Radio Shack is not too happy about this TI venture, could rack havoc with the Models 12 and 16. IBM has other things, down in the basement, just waiting for the right time. And DEC has a, believe it or not, name recognition problem with the general public. Too many potential users know the other names much better. DEC will go to TV to try to get that 'glitch' fixed, and soon.

Canadian Readers Note

For the past two or three months we have received complaints from some of you that as of last October (1982), you have not been receiving your 68 MICRO JOURNAL. This seems to be mostly in the following zip code areas: M4J - M4C - K0J - and a few others, but these mainly.

We have remailed, at our expense, additional copies. However, we cannot continue this. We have had our postal authorities examine our mailing and have verified that the bundles, for these areas, have been posted, each month. Which leads me to believe that the problem lie on the Canadian end. So, if you are having problems **PLEASE** contact **your** postal authorities. Also keep me informed, I will do all I can with our postal system, but I **MUST HAVE YOU COOPERATION!!!** Please inform them that we mail **2d Class Newspaper and Magazine,** which is the same here as first class (the fastest and not to be delayed, such as 'Bulk or Junk' mail). Again I repeat, it seems to be confined to the above postal zip code areas in Canada. I regret problems such as this and we will do all we can to get them resolved, but your assistance is necessary.

68 Micro Journal CBB

We have had our computer bulletin board up and running for 3 or so months now. It runs standard FLEX™ and has been received well. owever, we do have some **bugs,** and it may be awhile before we, and Ma Bell can get them all worked out. However, it seems that only a very small percentage of those who call and talk to the BBS have any trouble.

Our worst problem is that we are on a non-electronic switching system. Which actually means that the telephone substation, that we are hooked to is of an earlier vintage. This caused us to have lines that exhibit losses and noises beyond what other more modern systems experience.

It appears that if you are calling from an area that has 'clean' lines, then our lines, most of the time, will not be sufficiently bad to cause errors in data transfers. However, if both your line, and our line, are both lossy and noisy, then we have a problem.

It took us a couple of months to figure out what was happening. I call our system nearly every evening, to see who has called in and left messages and what have you.

Never have I experienced any problems, becau e I am, at home, on better lines. Now, down at the printing plant I get errors occasionally, there the lines, being in an industrial area, are 'dirty', hence, errors.

What this all boils down to is that until Ma Bell gets our new electronic station in, out here in Hixson (a semi-rural community), some of you will experience problems, to some degree or another, with our BBS. I can only apologize and hope that Ma gets with the program, and gets our telephone system changed over soon.

I guess that is the price we have to pay to be out here in the 'sticks'. But I can think of no greater pleasure that to be able to look out the window and watch the horses graze, or walk back into the woods about a hundred yards and fish for trout, crappie, catfish, or whatever, catch 'um to.

For you that have not checked in the telephone number is:

1-615-842-6809

We have a few text files and programs that you might like, and would appreciate any that you might want to leave for others. Please read our **INFO.TXT** and **README.TXT** files first, it may save some valuable telephone time. It is not the most sophisticated system, but it suffices. Give us a call sometime.

DMW - - -

COLOR User Notes

Robert L. Nay

5900 Cassandra Smith Rd.

Hixson, Tn. 37343

Due to the numerous requests I have received over the past several months for "more information on FLEX", or for explanations of different features of the FLEX Disk Operating System, I began a *general* discussion of Disk Systems and the FLEX Disk Operating System last month. This month I will complete the discussion of using FLEX on the Color Computer.

Last month, I discussed, in GENERAL terms, Disk Systems (the Computer, Disk Controller, and Disk Drives and how they work together), Disk Operating Systems (DOS) and how they work, and the Disks themselves. We left off with a Formatted Disk which had been "MAKESYS'd" to allow it to be used as a SYSTEM Disk that would 'boot' (load and execute) the FLEX Disk Operating System with an entry of RUN"FLEX" from Color Computer Disk BASIC. I will use the DATA-COMP Version of the FLEX Conversions for the Color Computer for this discussion, but you will have no problem with the few differences found in the Frank Hogg or Atomtronics conversions (I have not seen any of the other conversions, so I don't know how close they follow the normal FLEX Standards, or how they make some of the special Radio Shack to FLEX or FLEX to RS changes).

SYSTEM Disk Files

There are several types of Files (programs) that we want on the System Disk. First, the "SYSTEM Files" (ones with a .SYS Extension) MUST be on the System Disk. Next, we will want all of the "COMMAND Files" on this Disk (those with a .CMD Extension). Finally, we want various other Files on this Disk that are either required by some of the Command Files, or are ones that we will want during the normal use of our Disk System. We will normally use the COPY.CMD to put these files on our System Disk, so let's look at it first.

COPY.CMD

The FLEX COPY.CMD Utility is one of the more flexible and most used Files on the Disk. It can "copy" one File, several Files, or ALL of the Files, from one disk to another with a single Command Entry (but you MUST have at least TWO Disk Drives -- that is why most of the FLEX Conversions for the Color Computer provide a *Single* Disk Copy Utility). Assume we have 'assigned' the SYSTEM Disk Drive to be Drive 0 and the WORK Disk Drive to be Drive 1 (with the ASN.CMD, which will be discussed later). The simplest use of the COPY Command would be an entry of

```
COPY 0 1   -or-   COPY,0,1
```

This would Copy ALL of the Files from the Disk in Drive 0 to the Disk in Drive 1.

Let's drop back and repeat some of last months background information again. Remember, I said that the normal FLEX Command consists of a Drive Number, a Filename, and an Extension, all separated by periods. The previous Command, fully expanded, would have been

```
0.COPY.CMD 0 1
```

(or COPY.CMD.0 0 1 --- The Drive Number can be either in front of, or after, the Command; if an Extension is provided or required, it MUST follow the Command). Also, remember that this Command is entered from the FLEX System prompt of three Plus Signs (+++). We only need to enter the single word "COPY" because FLEX is EXPECTING a Command; it has given us the '+++' prompt, asking "What do you want me to do next?". It then EXPECTS the Command we gave it to have a '.CMD' Extension, AND it expects to find THAT Command on the SYSTEM Disk. If we had assigned the System Drive to be Drive 1, it would then have looked on the Disk in Drive 1 for "COPY.CMD". That is the reason we want all of the Files with a '.CMD' Extension to be on the System Disk in the Drive we have told FLEX is the 'System' Drive; we can then simplify our Command entries to single word Commands. But, we DO NOT have to have a '.CMD' Extension to be able to use a File as a Command; the only requirement is that the File be an operable BINARY Program. We only have to supply the Extension IF it is NOT '.CMD'; we could use COPY.BIN, COPY.TST, COPY.TRY, etc., whatever the Extension of the File happens to be, by including it in the Command Line.

Now, back to COPY.CMD (with the assumption that we have assigned System = 0 and Work = 1). The simple example we started with of

```
COPY 0 1
```

is often used to make 'Backups' of a Disk (where the information is valuable, and we don't want to risk losing it). COPY can be used in ANY situation with ANY Disk FORMAT combination! We can COPY from Single Density to Double Density or vice versa, from Single Sided to Double Sided, etc. One of the BEST features of FLEX is that the ONLY time we need to tell it the Disk configuration we are using is when we format the Disk with NEWDISK, FORMAT, or what ever Command *your* System uses. The FLEX DOS can determine the Disk Format rom the System Information Sector (Track 0, Sector 3, normally from Byte Numbers 38 and 39 -- Re. "FLEX *Diagnostics* Manual", p. 35) and operate accordingly. With the DATA-COMP and Atomtronics FLEX Conversions, you don't even need to tell FLEX whether the Drives are Single or Double Sided, how many Tracks they have, etc.; if you try to read a Double Sided Disk in a Single Sided Drive, you will probably get a "DISK READ ERROR" message, but no Files will be damaged. A Write to a Double Sided Disk in a Single Sided Drive will often result in "DISK WRITE ERRORs" and you will need to "Delete" that File, but this is also usually not 'fatal' (but ALWAYS use caution when writing to Disks). A Single Sided Disk in a Double Sided Drive works normally for both Reads and Writes.

Another feature of the COPY.CMD is that it will "clean up a Disk". If you have worked with a Disk a while, you end up with numerous Deletions and Additions. Many times the File you delete is smaller than the File you copy back onto the Disk. FLEX uses ALL of the Disk; if there is only a couple of Sectors here, and some more somewhere else, it will use them in pieces until the Disk is full. This leads to 'fragmented' Files; i.e., they are spread all over the Disk. The only problem with this is that it takes *longer* to Read and Write a *fragmented* File because the Read/Write Head has to be moved to so many different places. When you "COPY" a File, FLEX will place that File in sequence on the Disk you are copying too, providing faster Disk Reads and Writes on that Disk.

COPY.CMD can be used to copy 'specified' multiple Files with a single Command Entry. Suppose you wanted to copy ALL of the "Text" Files from Drive 1 to Drive 0. An entry of

```
COPY 1 0 .TXT
```

would accomplish this; any File on the Disk in Drive 1 that had a '.TXT' Extension would be copied over to the Disk in Drive 0. Or suppose you had several Files with names like PAG1.TXT, PAG2.TXT, PAG3.TXT, PAG.CMD, etc. A Command entry of

```
COPY 0 1 PAG
```

would copy ALL of the Files that begin with "PAG" from Drive 0 to Drive 1, which would include all of the ones I listed above (including the different Extensions). A Command entry of

```
COPY 0 1 P .TXT .CMD CA .SYS
```

would copy ALL Files on Drive 0 whose Filename begins with a "P", ALL Files with a ".TXT" Extension, ALL Files with a ".CMD" Extension, ALL Files whose Filename begins with "CA", and ALL Files with a ".SYS" Extension from Drive 0 to Drive 1. This makes "COPY.CMD" an extremely useful and easy to use Utility. (I'm sure you have become frustrated typing all of the information required in Radio Shack Disk Basic when trying to copy several Files from one Disk to another; I sure have.)

Finally, you can copy a SINGLE File by specifying the FULL Filename, as follows:

```
COPY 0.PAGI.TXT 1   -or-   COPY PAGI.TXT 0
```

The first example would copy ONLY the File PAGI.TXT from Drive 0 to Drive 1; the second one would copy ONLY the File PAGI.TXT from Drive 1 to Drive 0 (remember, we have 'assigned' System = 0 and Work = 1, so we don't need a Drive Number in the second example; FLEX will automatically look to the Work Drive for the File we want it to "Work" on). This version is handy if you have several Files that begin with the same letters, but you only want to copy ONE of them.

We can now see that a simple entry of

```
COPY 0 1 .SYS .CMD
```

will accomplish almost all of the File Transfers needed in making up a new System Disk, and the Disk Accesses will be faster on the new Disk because we have eliminated the "fragmentation". The final step is to transfer the rest of the Files you need onto your System Disk. These will normally consist of "Text" Files such as Dictionaries, "System Equate Files" for Disassemblers, "Execution Files", etc. One of the most important of the "Execution" Files is the "STARTUP.TXT" File, which tells FLEX about the 'configuration' of your specific Computer System. First, we will investigate the "EXEC.CMD", and then look at the "STARTUP.TXT" File.

EXEC.CMD

The "EXEC.CMD" executes a Text File which contains of a series of Commands. It is an extremely powerful feature of FLEX. You could set up a Command Sequence in the "STARTUP.TXT" File which would allow a computer operator with NO knowledge of how FLEX works to make one simple Entry (such as RUN"FLEX" from Radio Shack Disk BASIC) from the Keyboard, and by simply answering prompts, run Accounting Programs, enter the data, make Backups, etc. Let's look at a couple of examples (refer to the "FLEX User's Manual" for full details on the Commands in these examples).

I use an Epson MX-80 Printer which allows many "Escape Sequence" controls for various printing formats, etc. Often I need to Print a Listing in Compressed Print with 8 Lines to the Inch. I made up a File named "PRNTCMP8.TXT" with the "BUILD.CMD" as follows:

```
P HECHO 1B 50 1B 30
```

(HECHO.CMD is one of the Utilities in TSC's "FLEX Utilities" package -- refer to the end of this Column for a list and discussion of those Utilities.) Since I may want to use this File at any time, I have it on my System Disk (then it doesn't matter which Work Disk I am using, the File is available). I also "RENAMEd" EXEC.CMD to "E.CMD", so all I have to do to change my Printer to Print Compressed Text at 8 Lines per Inch is enter

```
E 0.PRNTCMP8
```

and it is done; I don't have to look up the codes or anything else. (Since I have 'assigned' W=1, I must specify Drive 0, or FLEX will look to the Work Drive for the "PRNTCMP8.TXT" File.) Simple, huh??

Let's look at a more complex example. When you use the "NEWDISK.CMD" to format a new disk on the DATA-COMP FLEX Conversion, it asks several questions. The "I.CMD" allows us to substitute a Disk File Input for the Keyboard Input. This is a 'VERBATIM' substitution; if you do not normally terminate an entry with a <cr> when answering the questions, DO NOT enter a <cr> in the "I.CMD" File. A File for the "I.CMD" which would Format a new disk with 35 Tracks, Single Density, with a name of "FMATERS", would look like this:

```
YNNFMATERS
0
35
Y
```

Notice that there is no <cr> after the "Y's" and "N's", you just hit a "Y" or "N" when answering the NEWDISK questions, so don't put a <cr>, space, or anything else between them in the "I" File; if the Input is a single letter, that's what the program takes from the "I" File. Now, let's suppose we named that File "FMAT35" for 'Format 35 Tracks'. We could then make up another File named "F35.TXT" on Drive 0 which would look like this:

```
I 0.FMAT35 NEWDISK 1
```

and when we entered "EXEC 0.F35", the Disk in Drive 1 would be "NEWDISK'd" (NOTE: make sure it is a Disk that you are SURE you want to NEWDISK, because you don't get a chance to back out once the EXECution begins). Notice that we need to tell FLEX where the question answers are coming from before we give it the Command when using this procedure (just like the "P.CMD").

Let me show you one more example; this one is much more complex, and will give you something to think about. I will not go into full details, except to note a couple of things. First, the "CONTIN.CMD" and "ECHO.CMD" is from the previously mentioned "FLEX Utilities" package, and "DISKNAME.CMD" is from a Utility published in '68' Micro a while back, which allows us to rename a Disk. The File "DSKNAM.TXT" is simply "FMATERS<cr>", and the File "FMAT35.TXT" is as shown above. The File shown below will be named "F35.TXT":

```
TTYSET PS=N                 (Turn off Pause)
ECHO FORMATTING 35T SS SD   (Tell me what it
                             is doing)
I 0.FMAT35 NEWDISK I        (NEWDISK the Disk)
ECHO "MAKESYS" 1            (Tell me about it)
MAKESYS I                   (Do it)
ECHO COPYING 0 TO 1         (Tell me about it)
COPY 0 1                    (Do it)
I 0.DSKNAM DISKNAME 1       (Name the Disk)
CONTIN                      (Pause and ask me
                  If I want it to Continue)
E F35                       (If so, do again)
```

The comments in parenthesis are not included in the File; they are explanations of what is going on. The "ECHO.CMD" causes the text following to be printed on the Display. The "CONTIN" asks if I want to Continue, and is answered with a "Y" if I do, or a "N" if not. Since the EXECution stops while it is waiting on a reply, I can remove the Disk that I have just made, install another Blank one, then hit the "Y" Key, and it will go to the next Command, which is the same "E F35" that I entered to start the procedure. When I am finished, I simply hit the "N" to terminate the procedure. One other note; the "DISKNAME.CMD" asks for a "Disk Name", which is entered and completed with a <cr>, and then it asks for a "Disk Number", which is also terminated with a <cr>. BUT, the "DSKNAM" File only has a NAME<cr> in it. This causes the System to wait for an entry FROM the KEYBOARD; I can enter a Disk Number, which changes with each Disk, and as soon as I hit the <cr>, the Program continues on it's way. This is the procedure we use in making up the DATA-COMP "FMATERS" FLEX Conversion Disks (although we have 3 Drives; Drive 0 has the FLEX System Disk with these Commands on it, Drive 1 has the "MASTER" Disk, and Drive 2 contains the New Disk to be made up: so I changed some of the Drive Assignments in the example above). As you can see, the "EXEC.CMD" and some of the Utilities provide the capability of making up complete "Job Command Procedures".

STARTUP.TXT File

Now that we understand the use of "EXECution Files", I will discuss another of the really NICE features of FLEX; the "STARTUP.TXT" File. When the "FLEX.SYS" File is initially loaded, the 'Boot Program' transfers control to the FLEX Program. It proceeds to initialize itself; check the amount of memory available, set up it's Tables and Variables, ask you for the Date, etc. (not necessarily in that order). It then looks for a File named "0.STARTUP.TXT", and proceeds to "EXECute" it, if found. (This means that you MUST have "EXEC.CMD" and any .CMD used in the "STARTUP.TXT" File on Drive 0 with "FLEX.SYS".) STARTUP.TXT contains a list of Commands that you would normally have to enter every time you loaded FLEX to get the Computer System set up for normal operation.

The "STARTUP.TXT" is normally made up with the "BUILD.CMD", which is a simple utility that allows you to make up small Text Files with very little fuss (a baby Editor). You do not have any editing capability, except 'Backspace', so it is not much good for a very long File, but it is simple and handy. You simply enter

```
BUILD 0.STARTUP
```

and begin typing when you get the Equals Sign (=) prompt. When you have finished, enter a Pound Sign (#) as the first character following the (=) prompt, and the File is saved to Disk with a ".TXT" Extension. I used the Drive Number to force "STARTUP.TXT" to be

saved to the System Disk instead of the Work Disk; otherwise, I would have to COPY it over to Drive 0.

Besides the "initial Setup" Commands, the STARTUP File is handy for leaving yourself messages. Suppose you have been chasing a "small bug" (aren't they all) in a Program you are writing, and you want to leave a reminder about where you were when you quit the session. If you include a Command like "LIST 0.HELLO" in the STARTUP File, and then make up another Text File named "HELLO.TXT" on Drive 0, it will be "listed" to the Screen the next time you load FLEX with the Disk Basic Command 'RUN"FLEX"'. Why another File? As I said, "BUILD" does not allow editing, so you have to either type everything again to change something in the File, or use a full-blown Editor like "EDIT.CMD" (TSC's Line Editor), "STYLO.CMD" (Control-C's 'STYLOGRAPH' Word Processor -- which I am using now), or "SCREDIT.CMD" (Alford Associates 'SCREDITOR III' Word Processor), to list some examples. Normally, once you settle on a configuration, you won't want to be changing it much, so just make up another File like "HELLO". If you don't want to leave a message, then you can either "DELETE 0.HELLO.TXT", which will then give you a "FILE NOT FOUND" Error Message when you boot FLEX, or simply "BUILD" another HELLO.TXT which consists of a single "space" (it must have SOMETHING, or it will not save a File when you exit BUILD).

Lets look at my STARTUP.TXT File, and see what one might look like.

```
V51X24:DISKRATE 0,0:ASN S=0 W=1:TTY DP=24 ES=31:ASN
```

(Whoa, I know you don't have a "TTY.CMD"; I'm lazy and 'RENAME'd' TTYSET.CMD to TTY.CMD.) Before we get into each of the Commands and what they do, notice the colons: you can give FLEX a whole list of things to do in one Command Line (before you hit the ENTER Key) by separating the different C mands with the colon. Also, you will notice that I use the 'space' instead of the 'comma' as a separator; it's a lot easier to hit on the Keyboard.

V51X24.CMD

The first Command is the V51X24.CMD, which brings the System up in the 51 column by 24 line Display Screen (with Lower Case letters). I always use it, so rather than having to begin each session with the Command V51X24, I installed it in the STARTUP File. Also, with the DATA-COMP FLEX Conversion, if you want to use this, or any of the special Display Screens in the STARTUP File,

It MUST be the FIRST COMMAND in the LIST.

Other than this one item, everything else can be in any order you please.

DISKRATE.CMD

The "DISKRATE.CMD" (a DATA-COMP Color Computer FLEX Conversion Utility) tells FLEX how fast your Disk Drives can 'Step' between Tracks. Since I am using the new Tandon TM-100 Series Disk Drives, I set the 'Stepping Rate' at the fastest value with the Command "DISKRATE 0,0". You can tell when this Command takes effect, because the 'Stepping' goes from separate 'clicks' to a 'buzz'. I can't honestly say that I can tell a difference in the Read/Write Access times for any single Program, but I sure can tell the Drives are stepping quicker when they go from the Directory to the File on the Disk.

The Frank Hogg Color Computer FLEX Conversion includes this in their "SETUP.CMD" Utility, along with the Number of Tracks on the Drive, a begin 'Write Precompensation' location, setting the "logical" Drive Number, and whether they are Single or Double Sided and Single or Double Density capable Drives (the DATA-COMP and Atomtronics Conversions do not worry about most of these 'Drive Characteristics'; they read the appropriate information off of the "System Information Sector" of each Disk, and react accordingly; or expect the Computer Operator to know what type of Disk Drive is in which location, and what Disks he can use in which Drive).

ASN.CMD

I have discussed the ASN.CMD indirectly throughout this discussion, so you already have a pretty good idea of what it is and what it accomplishes. All "ASN.CMD" does is tell FLEX which Drive is your SYST M Drive and which is your WORK Drive. You can change assignments any time you want, and they do not have to be different Drive Numbers.

Also, you can use an "A" in place of a number to assign "ALL" Drives as a System or Work Drive, as follows:

```
ASN S=A
```

This would make ALL of the Drives a System Drive; FLEX would look on Drive 0 for a Command, and if it did not find it there, look on Drive 1, etc. This is used quite often if you have a Hard Disk in the Disk System, because it can hold all of the Commands you could ever want. We normally work with an assignment of S=A,W=1 at '68' Micro where we have a pair of 8" Drives on 0 and 1 and a Hard Disk on Drive 2. We leave the "Write Protect" ON on the Hard Disk and use Drive 1 as a Work Disk. When we have a File "finalized" on the 8" Drive, we then turn the "Write Protect" OFF and transfer it over to the Hard Disk, then get the "Write Protect" back ON as soon as possible. This provides as much safety as possible for all of the information on that unit. (As I said last month, the greatest chance for destroying a Disk is when you are writing to it; with "Write Protect" normally ON, even a Power Surge or Power Failure can not cause something to be written to the Hard Disk. Why would a "spurious" Write to a Disk cause so much damage? The last thing FLEX does when something is written to a Disk is update the Directory. This leaves the Read/Write Head located at Track 0 on the Disk, and spurious Writes wipe out the Directory. With a messed up Directory, FLEX can not locate a File on the Disk, even though that File will probably be all right. The "FLEX Diagnostics " package from TSC provides several Utilities and a lot of good information for repairing damaged Disks -- see Discussion at the end of this Column.)

TTYSET.CMD

"TTYSET.CMD" lets FLEX know how your Terminal (or Keyboard/Video Display System) and Printer are set up, so it knows how to control various Displays, Printer outputs, etc. As I stated before, I have renamed TTYSET.CMD to TTY.CMD. An often used Command is

```
TTY PS=Y
```

which turns the "PAUSE" feature back on after it has been turned OFF by the use of the Printe , etc. TTY is just easier to type than TTYSET. The "PAUSE" is controlled with a Y for "Yes" or N for "No". If you CAT or LIST a File with "Pause" off, everything scrolls right on by, requiring you to hit the "ESCAPE" Key to stop it.

If you simply enter

```
TTYSET
```

you will get a listing of all of the parameters controlled by TTYSET.CMD, and what they are at the present time. Many are shown as Hex Values, others are Decimal Numbers.

The TTYSET parameters that are relevant for the Color Computer FLEX Conversions, and some of the settings, are as follows:

"BS" is the Hex Code for the Backspace Key: $08 is the Code put out when you hit the "Left Arrow" on the Color Computer Keyboard, which is the Standard for the Backspace.

"DL" is the Code to "Delete this Line"; $18 is a "Control X". It is used if you are entering a Command from the FLEX Prompt and decide that is not what you want to do after all. You can wipe it out with Backspaces; but it is easier to hit a Control X, which 'deletes' everything on that Line and FLEX answers with three Question Marks instead of Plus Signs.

"DP" is the number of lines you want displayed per 'page' on the screen during listings. I have mine set at 24 because I use the 51X24 Display Screen; with 'Pause' ON, a CAT, LIST, etc., will list 24 lines and wait for either and "Escape" to continue the display, or wait for a <cr> (the ENTER key) which causes the listing to stop and the System returns to the FLEX Prompt. If you are using the 32x16 Display, set "DP=16" to get a full Display Page at the time.

"WD" sets the width of the Display Screen; set to 0 for the Color Computer's Video Display System.

"PS" is the Pause Control, which I have already discussed.

"ES" is the Escape Key definition. This is normally $1B for Escape (the redefined Shift <BREAK> Key on the Color Computer); I have mine set at $31 for the "Number One" Key because I am used to hitting the Upper Left Key on the Keyboard for Escape (the normal location on Terminals). This presents no problem, because it is only examined for listing stops and starts, and functions normally when you might be entering the actual Number "ONE". FLEX also allows

you to "Pause" a Printer Listing by hitting the defined Escape Key; for example, if you have begun a Print Out and want to stop it, hit "Escape" and wait for the Cursor, then hit <cr>, and the Printing will be stopped and the System returns to FLEX. If "Pause" happens to be off (say you have just finished a Printer listing, which turns "Pause" OFF) and you call for some type of Display Listing before you turn "Pause" back ON, you can hit "Escape" and pause the listing, hit Escape again to continue it, or hit a <cr> anytime you have a Cursor in the Pause mode to return to FLEX.

The rest of the parameters should be set to 0 on the Color Computer FLEX System. The best procedure is to enter TTYSET<cr> as soon as you get FLEX up on the Color Computer to see what the TTYSET default settings are, and change the ones you want changed with the "STARTUP.TXT" File. Then, each time you bring FLEX 'up', they are set for you.

The final "ASN" shown in my STARTUP.TXT File simply prints out the System Drive Assignments on the Screen when FLEX has completed it's initialization. Why print it out?? Habit, I suppose; I just normally set it up that way.

That about completes the discussion on the FLEX Disk Operating System. This information should help you "get off of the ground" with the DOS if you have FLEX, or show you how easy it is to work with if you have been wondering about all of this "FLEX Stuff" I have been talking about the past several months. As I have stated many times before, FLEX allows us to make some real USE of the COLOR COMPUTER; when you are running the FLEX DOS, you can have a WORKING Computer, rather than a "game machine".

TSC "Utility" and "Diagnostic" Programs

I have discussed many of the numerous Programs available that run under the FLEX DOS in the past, and will discuss more in the future. Let me mention a couple of TSC's Utility Software Packages for the FLEX System; "FLEX Utilities" and "FLEX Diagnostics". These are extremely useful if you are seriously using the Color Computer and the FLEX Disk Operating System. You can obtain them through most FLEX Software outlets (DATA-COMP, Frank Hogg, etc.) or directly from TSC.

"FLEX Utilities" consists of 17 Utilities along with the Source Code for them. The Utilities are:

CHECK.CMD - compares two (Text or Binary) Files to see if they are the same.

CMPMEM.CMD - compares a Disk Binary File with the contents of the memory locations where it would normally be located.

CONTIN.CMD - used in Text "EXEC" Files to cause the execution to halt and ask whether you want it to "continue" or not. A very handy utility. Reference the previous "EXEC.CMD" discussion.

DIR.CMD - similar to CAT except you get more information. I use it for a Printer output of the Disk Directory (It uses almost 80 columns per File, and tells you how many Files you have on the Disk, the size of the largest File, how many Sectors you have used, and how many are left).

DUMP.CMD - displays a Disk File one Sector at the time on the Display, in both Hex and ASCII.

ECHO.CMD - another Utility for use in "EXEC" Files. This allows you to put a 'note' in the Text File you are executing that will be displayed on the Screen. If the procedure is a long one, you can use it to Print a message on the Display that would advise the User of what is going on, such as "Please stand by a moment while I load the Program", or "This will take about 10 minutes; why not have a cup of Coffee while I work on it?", etc. Refer to the discussion on "EXEC.CMD" for other examples.

EXTRACT.CMD - used to develop a new Text File by extracting Text from OTHER Text Files. This is a powerful and useful Utility.

FILES.CMD - similar to CAT, but displays ONLY the Filename and Extension. Provides a short, quick look at a Disk Directory.

FIND.CMD - another very handy Utility; FIND will locate the lines containing any specified "string" in a Text File on Disk without your having to load the File into an Editor, etc. For example, "FIND,CHAP1,16 BIT MICROS" would list each line in the File named "CHAP1.TXT" that contained the words "16 BIT MICROS".

FREE.CMD - a quick way to find out how many Sectors are left on the Disk. Just like the Radio Shack "?FREE(0)" Disk Basic Command.

HECHO.CMD - another goody; like "ECHO" above except it passes a Hex Value. I use it in short EXEC Files to set up the Printer for different Print Fonts, etc. See the "PRNTCMP8.TXT" example above.

MAP.CMD - shows the Load and Transfer Addresses of Binary Files. You can determine where a Program will load into memory and where the Execution Address is. One very good use for this Command is in determining where the different Utility Programs load, because if it does not load into the $C100 Utility Command Space, it can not be used by the different Programs that allow you to "Pass a Command to FLEX", such as "EDIT.CMD" or "STYLO.CMD". Commands like "COPY", "EXEC", "SAVE.LOW" load into lower RAM, and will wipe out the Program you are running.

MEMEND.CMD - used to either determine or set the FLEX Memory End.

PDEL.CMD - a very useful Command for cleaning a Disk or removing several Files, instead of just one when using the "DELETE.CMD". PDEL stands for "Prompting DELETE"; it lists the File to the Display and asks if you want to Delete it. You hit the "Y" Key to Delete it, or the "N" Key to leave it on the Disk and go to the next one. PDEL also uses the "Wildcard Filename" capabilities of the CAT or COPY Commands; you can enter

PDEL .TXT CA

and it will list each File with a ".TXT" Extension on the Work Drive and ask if you want it Deleted, and then list each File whose Filename begins with "CA", and ask if you want to Delete it. This one Utility is almost worth the price of the whole Package.

RUN.CMD - used to load and optionally execute a Position Independent Code Program at a different location in memory.

SPLIT.CMD - used to "SPLIT" a Text File into two different Files. The original File remains unchanged.

ZAP.CMD - used to DELETE all Files matching a "match list". For example, "ZAP .BAK" would delete ALL Files on the Work Drive with a .BAK Extension. No warning is given, they are GONE!!

All in all, a VERY handy set of Utilities to have around. Some of these make the use of EXEC Files very powerful, some provide useful information, and some make the FLEX DOS easier to use.

The "FLEX Diagnostics" Package of Programs provide 17 Utilities that allow you to Test Memory and Disks, recover "Deleted" Files, etc. You also get an EXCELLENT Manual with a lot of good information on the Structure of a FLEX File on the Disk, different Memory Test procedures, different methods to use in salvaging a damaged Disk, etc. Again, if you are seriously using the Computer, and have important Files on Disks, this set of Utilities is a MUST.

The "Diagnostic Utilities" include the following Programs:

MEMORY TESTS

CONVERGE.CMD - primarily for detecting shorted address and data lines.

DYNAMIC.CMD - checks bit drop out due to refresh timing problems.

QUICK.CMD - a quick check of a block of memory.

RANDOM.CMD - tests a block of memory using pseudo-random bit patterns.

WALK0.CMD - a "walking zero" memory test routine.

WALK1.CMD - a "walking one" memory test routine.

DISK DIAGNOSTIC PROGRAMS

TEST.CMD - reads each Sector, reporting those with errors.

VALIDATE.CMD - checks a FLEX Diskette for structural errors caused by hardware or software problems.

FILETEST.CMD - tests the Diskette by Files for errors. Can also be used for checking Boot and System Sectors, Directory Sectors, Free Chain, etc.

DATA RECOVERY UTILITIES

RAWCOPY.CMD - copies a file, ignoring checksum errors whenever possible. Once you get a "readable" copy, you can use "EXAMINE.CMD" or other routines to try to reconstruct a bad Sector of information.

REBUILD.CMD - attempts to find files on a crashed Diskette whose Directory has been destroyed, and copy them to another Drive.

RECOVER.CMD - copies Files to another Drive from user supplied Track and Sector information.

UNDELETE.CMD - HANDY; this Utility allows you to recover a File that you have "DELETEd" by mistake.

MISC. DISK REPAIR UTILITIES

COPYR.CMD - used to restore the file s ctor map after using "REBUILD", or to put a file sector map together for any sequential file.

FLAW.CMD - used to remove bad sectors from the Free Chain.

EXAMINE.CMD - one of the 'most used' Utilities. EXAMINE allows you to read, write, and/or modify ANY Sector on the Disk (similar to the DATA-COMP "DISKEX.CMD" Utility).

The only problem is that a couple of the Disk Repair and Examine Utilities do not know about Double Density 5 1/4" Disks, and the Memory Test Routines need some "pointers" changed, but there is information in the Manual which will allow you to make the necessary changes. Maybe I can find time to work it out and pass it on if there is sufficient interest.

Finally, back issues of '68' Micro Journal are a GOLD MINE of information and Utilities for the FLEX Operating System. Most issues have at least a couple of handy Utilities, normally in the "Bit Bucket" Section. Just be aware that these are published JUST EXACTLY like we received them, and you may need to change them slightly, or contact the author, to get them to run on YOUR Computer System.

--- RLN ---

PENGUIN Sys

A Review of the Penguin Business Systems
Service Bureau Applications Software

by E. M. (Bud) Pass, Ph.D.
Computer Systems Consultants, Inc.
1454 Latin Lane, Conyers, GA 30207
Telephone Number 404-483-1717/4570

GENERAL SYSTEM DESCRIPTION

The Penguin Business Systems Service Business Applications Software is an online, interactive accounting system which functions through a group of data files managed with a data base system. It includes the following subsystems:

Data File Management System
Accounting Management System
Checking Account Management System
Payroll Management System

The individual subsystems are mutually independent except for the shared use of the data base files. This independence allows the staged or partial implementation of the system and the use of utility data base functions on any of the files used by the subsystems.

The system is menu-driven and prompting, although it is does not use cursor control, and hence is not full-screen oriented.

The user interface, the hardware and software requirements, and an evaluation will be presented below.

SYSTEM MENUS AND DESCRIPTIONS

In order to provide an indication of the capability of the system, the system and subsystem menus are provided below. Each has a description of the system or subsystem and possible operator actions at each stage.

SERVICE BUSINESS SYSTEM
Version 4.01

Copyright (C) - February 15, 1982
All rights reserved by:
Penguin Business Systems
1129 South Florence Avenue
Tulsa, Oklahoma 74104
(918) 592-1227

1) Disk File Manager System (DFM/09)
2) Accounting Management System (AMS)
3) Checking Management System (CAMS)
4) Payroll Management System (PMS)
5) Log off the SBS System

Your Choice.... ?

The system menu provides the initial and final control points for the system and connects the subsystems.

* DISK FILE MANAGEMENT SYSTEM
* Version 4.05
*
* All rights reserved by:
* Penguin Business Systems
*

1) Add a record to a file.
2) Change a record in a file.
3) Kill a record in a file.
4) Remove a record from a file index.
5) Restore a record to a file index.
6) Label generator.
7) List a file.
8) Create a new file.
9) Sort a file.
10) Backup the file.
11) Return to SBS control.

Select from the above ?

The Disk File Management subsystem is part of a data base management system, which is the basic building block for the Account Management subsystem, the Checking Account Management subsystem, and the Payroll Management subsystem. This subsystem provides direct utility functions in support of the remainder of the subsystems.

Operator input in this (as in all) subsystems is verified before any permanent changes are made to the data base files. All data base files are backed-up by this subsystem.

ACCOUNTING MANAGEMENT SYSTEM
Version 4.02

Copyright (C) - February 15, 1982
All rights reserved by:
Penguin Business Systems
1129 South Florence Avenue
Tulsa, Oklahoma 74104
(918) 592-1227

1)	Sales & Income	8)	Day Ending
2)	Purchases	9)	Month Ending
3)	Receivables Inquiries	10)	Quarter Ending
4)	Payable Inquiries	11)	Year Ending
5)	Journal Entries	12)	Current Payables Rpt
6)	Receivables Report	13)	Return to SBS Control
7)	Financial Reports		

Your Choice.... ?

The Accounting Management subsystem is a double entry, accrual method, bookkeeping system. It includes the functions normally performed by the following accounting procedures:

General Ledger
Accounts Receivable Ledger
Accounts Payable Ledger
General Journal
Receipts Register
Disbursements Register
Sales Register
Purchases Register

The subsystem is supplied with a sample universal general ledger chart of accounts which must be customized and completed with actual balances before use. It will support up to seven departments per company. It provides an audit trail in terms of a dual transaction numbering and dating scheme and the account referencing scheme. It produces ledgers, journals, and other reports on demand.

Checking Account Management System
Version 4.04

All rights reserved by:
Penguin Business Systems

1) Issue check
2) Deposit

```
3)     Activity fees
4)     Record cancelled check
5)     Reconcile Checkbook
6)     Return to SBS Control


Current Balance = $$$$$.$$


Which routine ?
```

The Checking Account Management subsystem is a check register system. It includes the following functions:

- Check Printing
- Deposit Recording
- Bank Activity Fees
- Payable Account Payments
- Check Cancelling
- Check Reconciliation

The subsystem maintains a checkbook data base and is interactive with the Accounts Management subsystem. It uses preprinted, prenumbered, fanfolded, checks assumed prepared on a dedicated correspondence quality printer, such as a Centronics 737 or 739. The current balance of the single checking account is displayed in the menu and is limited to ten million dollars, as is any single check or deposit.

```
------------------------------------------------------------------------

........................................................................

                     PAYROLL MANAGEMENT SYSTEM
                           Version 4.01
------------------------------------------------------------------------
                 Copyright (C) - February 23, 1982
                       All rights reserved by:
                      Penguin Business Systems
                      1129 South Florence Avenue
                        Tulsa, Oklahoma  74104
                           (918) 592-1227
........................................................................

  1)     Produce payroll              5)     Close the year
  2)     QTD report                   6)     Build new tax table
  3)     YTD report                   7)     Return to SBS control
  4)     Close the quarter

                       Your choice.... ?
------------------------------------------------------------------------
```

The Payroll Management system is a payroll manager. It operates its own payroll data base and is interactive with the Accounting Management subsystem and the Checking Account subsystem.

The subsystem provides quarterly and annual reports; other reports are generated by the Disk File Management subsystem. It allows for weekly payroll deductions and excess federal income tax withholding. It uses the same checks as used by the Checking Account Management subsystem.

SYSTEM HARDWARE/SOFTWARE AND OTHER REQUIREMENTS

Following is the minimum hardware/software configuration recommended for the use of the system:

SWTPC S/09 computer with 192K bytes RAM,
MP-S2 serial interfaces in ports 0 and 7,
DMF-2 with dual 8" DSDD floppy drives,
CDS-1 20M byte hard disk drive,

Centronics 737 or 739 printer with parallel interface,
Centronics 704 or Epson MX-100 serial 132 character printer,
CT-8212 or CT-8212W serial terminal,

TSC UniFLEX Operating System,
TSC UniFLEX Extended BASIC Interpreter.

Similar equipment may be substituted. Penguin Business Systems is a SWTPC dealer, so the SWTPC orientation of the configuration is understandable. The serial printer and terminal are assumed to be run at a speed of 1200 Baud. Since no cursor control is used by the system, any terminal supporting 80 columns and 24 rows may be used. The system will run on a system without hard disks with major speed degradation, especially when loading BASIC programs, but also when scanning the data base.

The system is written completely in BASIC for the TSC UniFLEX BASIC Interpreter. The system includes almost all source programs and all required files already set up for user customization. A demonstration system is available to registered SWTPC dealers for $320.00. The complete system retails for $3872.00, which includes one year's maintenance.

EVALUATION

The demonstration system apparently performs as specified in the manual. The system requires no terminal customization, as most competitive systems do, since it uses no cursor control. It is oriented toward a hard disk system and is somewhat tricky to set up for a non-hard-disk system, but it can be done with some difficulty. The demonstration system concept is a good marketing strategy, but requires very simple installation to be really effective. A minor complaint which detracts from the professional appearance of the system is that there are several mis-spellings and grammatical errors in operator messages.

The manual is complete and comprehensive as a reference manual. It is over 160 pages in length. It is not a user guide, nor is it a demo guide. there is no quick index nor cross-reference on how to perform a given common task quickly. This makes for much page-flipping and searching for new users, since the menu verbiage is necessarily terse. The manual is printed using a dot-matrix printer, and some pages in my copy were difficult to read. A manual printed using a daisy-wheel printer would have looked far more professional. Again, there were several mis-spellings and grammatical errors in the manual.

Supported by the Accounting Management subsystem, the general ledger chart of accounts is non-standard and somewhat restrictive, allowing only one company per system and only seven departments per company. The restriction of one company per system could be ameliorated somewhat thru the use of multiple directories on a hard disk or thru the means of running the system using multiple floppy disks for data base storage. Neither is as satisfactory as the direct processing of multiple companies per system and an unlimited number of departments per company.

Since the Accounting Management subsystem uses a four-digit account, the first digit is used for major divisions, and the second digit is used for minor divisions, most detail accounts will be restricted to 100 or 200 accounts per minor division. This type of restricted allocation will cause problems if a company has more than 100 payable or receivable accounts or 200 inventory accounts. Most modern general ledger systems use a six digit or longer system to avoid the number allocation problems associated with three or four digit chart of account systems.

Although the Accounting Management system supports accounts receivable and accounts payable journal entry, it supplies only only inquiry and reporting capabilities. It does not print aged receivable invoices nor does it print payable checks or vouchers.

The Checking Account Management subsystem has several potentially major problems, in addition to the restriction of one company per system. The primary one is the restriction of a company to one checking account. Many companies, even small ones, posess and use multiple checking accounts, and this restriction would imply that only one account could be managed with this subsystem. The other checking accounts would be managed manually and entered as manual journal entries.

the Payroll Management subsystem also has several potentially major problems, in addition to the single company problem. The major assumption which may cause problems is that all employees will be paid weekly and that insurance will be deducted monthly. Also, only one state tax deduction table is allowed and no local payroll tax deductions are allowed.

The major inconvenience with the use of the system concerns the method of data entry. No cursor control or formatted screens are supported, using the terminal as a glass teletype. Virtually all competing systems support formatted screen input although of varying sophistication. The Penguin system outputs a key word or phrase for each input desired. The operator is expected to respond to the prompt on a one for one basis. At the end of a group of input, the operator is asked if the input was OK. If it was not OK, the operator is required to re-enter the entire group of input. Compare this to formatted screen input in which the operator is allowed to correct any field on the screen by moving the cursor to the appropriate field and correcting only the offending characters.

SUMMARY

The Penguin Business Systems Service Business Applications Software apparently functions as documented. If the limitations are unimportant to a particular customer, it may be of interest as a candidate system. It is available from the following company:

Penguin Business Systems
1129 South Florence Avenue
Tulsa, Oklahoma 74104
Telephone 918-592-1227

1/9/83

Dear Don,

Reference the review for our Service Business Applications Software. There have been considerable updating and typo repairs made to the code as well as the manual.

1. Mis-spellings and grammatical errors have been corrected with the last 4 revisions. We are currently selling Version 4.05, and are constantly revising and correcting errors.

2. Please note that the correct title of the software is **Service Business Applications System.** It is not geared toward a service bureau, but rather a

service business, such as Plumbing, Heating, Air Conditioning, etc. Therefore, only one company per ledger.

3. With our latest revision we support **Aged Accounts Receivable Report, Open Accounts Payable Report** and **Accounts Payable Check Printing.**

4. This system was designed for single, small service business types, therefore, only one company on the system. With the recent SWTPC Mini-winies there is no 20 Mb requirement. It must be pointed out that the system maintains a customer record FOREVER.

One final comment: the price is $2,800 and the demo kit is $280.00.

Sincerey,

Jim Carter
Penguin Business Systems

"C" User Notes

Norm Commo
3 Pryor Road
Natick, MA 01760

Have you ever walked into a room to do something and said "Well... where do I start"? I have that feeling right now. Luckily, I made a commitment to get an initial feel for Middle C from Word's Worth so at least I have a direction. This will be a short month never the less while I reestablish my bearings for the column.

First off, let me say thanks to all of you who took the time to write with suggestions or questions. It really helps me find where I have done a good job and where I have missed the mark. Again, thanks!

I got a letter from a reader in England who asked a question that may be bothering a lot of you. In the last two columns, I used names like s_name and then would make references to "a struct of type s_name". He didn't quite understand what I meant by that phrase. So let's try to make it a little clearer.

All variables in C are declared. When you declare them you give their name and their type. Examples would be

```
char var1, var2;
int var3;
```

Here var1 and var2 were declared to be character variables. Structs are user defined, and in any given program as many as needed may be defined. So you can't just tell the compiler

```
struct var_name;
```

You might, after all, have more than one type of struct defined. That is why you use the form

```
struct s_name var_name;
```

This tells the compiler that you are declaring a variable (whose name is var_name) that is a struct that has previously been defined and has the "tag" s_name. So you see, "struct" is a data type just like "int" or "char". The only difference is that you can define different types of structs, so you must give each one a unique tag (name) and then refer to that definition using the name you gave it.
This is show below.

```
/* the definition of a struct */
struct tag {
          int var1;
          int var2;
          char strng[10];
/*        };
 * the declaration of a variable (var_name)
 * of type "struct tag"
 */
struct tag var_name;
```

Of course, like every rule, this one has an exception. If you are going to declare a struct variable that will not be used again, then you don't have to use a sturct "tag", but only the variable name. As in

```
/*
 * declaring a struct and a variable
 * at the same time
 */
struct  {
        int var1;
        int var2;
        char strng[10];
        } var_name;
```

This is a one time declaration. You can't declare another variable by referring to this definition because you didn't give it a tag.

Finally, you could do both by the declaration

```
struct tag  {
            int var1;
            int var2;
            char strng[10];
            } var_name;
```

With this format, you can declare other variables of the same type.

Don't forget. The definition or declaration of structs are bound by the rules of scope. So that if you want to use a struct globally, you must have declared it globally. This is usually what's done anyway. Also, in all the previous examples, "tag" "var name" and "s name" are just symbolic. You may use any name that you want. By the way, some of you expressed puzzlement over my term "boiler plate". That is just slang term for "template".

MIDDLE C REVIEW

With the release of Middle C, Word's Worth has elevated their compiler to be a proper subset of full C. Syntactically, it is a lot cleaner now, and they have added more features to boot.

Their compiler has a number of refinements and new features such as

Supports of all the binary operators except the)>= and ((=.

The proper logical operators && and ||.

Compiler recognition of the escape characters \b, \f, \n, \r, \t and \ddd.

The data type short (8 bits).

Decimal, octal and hexidecimal constants.

The compiler directives #if, #ifdef, #ifndef #endif.

A new and better optimizer, and a preprocessor.

RLOAD has been improved.

The variables for a function call are now loaded from right to left (the proper way), and the variable stack is now U instead of S.

The compiler runs faster and produces better code.

The runtime package now parses the command line passing in arguments and allows I/O redirection.

There are lot more functions supplied with the package.

I have run the compiler on two programs. One is the now (standard?) Eratosthenes Selve program. It compiled without difficulty and ran in 42 seconds on my 1Mhz system. For fun I assembled the unoptimized code. That version ran in 72 seconds.

The other program was the program date.c, which is included in this article. This is a program that takes as its arguments a date and returns the day of the week. Not particularly useful unless you have forgotten what day of the week you were born on, but it did have a few complex binary operators and it shows a technique that I use to load up arrays.

The program uncovered a couple of bugs. The problem was with the way I initialized the arrays. The

call to init array() with all the months is split into two lines. This is supposed to be perfectly legal. So, with fear and trepidation, I put it all on one line. I was worried that the compiler's line buffer might not be more than 80 characters. It didn't, until later.

The last line of date(), is a printf() statement. Originally, the variables sd, sm, and sy were called sday, smonth and syear respectively. I had to shorten them because the preprocessor gagged on that line and emitted a "line too long" error message. What is confusing is that this line is much shorter than the call to init array() that I had to change.

Once I made the changes the program compiled flawlessly and run properly. Compiling programs has been made a little cleaner too. You may specify everything that you want to do right on the command lines to the various passes. About the only complaint that I had in this department was that the various passes are too protective. If you name an output file that already exist, you are reminded of the fact and returned back to FLEX. I don't like those kind of programs. If I'm that dumb, then I deserve it! Let me learn the hard way, I'll be better off for it in the long run.

To get around this I designed the program c.txt, also included with this month's column. This is an assembler program that runs in the FLEX utility area and calls FLEX as a subroutine (at $CD48) to invoke all the passes necessary to get from source code to binary.

You call the program with a single parameter, the file name of the C source code. It will assume a default extension of ".C". The program then does the rest for you. The command line that will be handed over to FLEX is printed at the terminal and you are asked if you want to continue. If you say yes, then deletes the output file for that pass before invoking the pass. If you say no, then it returns control to FLEX.

I was going to do it in C, but I was afraid that I might have run out of room in the utility area. There was also the fact that I would most likely have ot pare down a lot of the overhead from the runtime package, figure out how to really use RLOAD to put the program and its data areas in the utilit area, etc, etc. Since I was in a rush I decided to brute force it with assembler.

It's certainly nothing spectacular, but it works and it DEFINITELY make the compilation a lot easier and more enjoyable.

As for the final rating of Middle C. I give it AA. I consider the bugs that were encountered to be rather severe so I can't quite give it AAA; but it's still one of the best buys around! I have also been very impressed by Word's Worth's integrity. Some rather nasty bugs got into the FLEX version when another programmer made the conversion from the SSB version. Within a few weeks, a new disk arri d with an update, at no cost.

WRAPUP

I want to end up by saying a few more words on date.c. This program was pulled right from Kernighan and Ritchie. The only change was from using initialized arrays to initializing them with something of a trick.

There are two arrays of int's and two arrays of pointers to char's. The arra s are initialized by a call to init array(). The compiler puts all the values on the stack in the call to the function. This is especially efficient with an array of pointers to char's. Let the compiler do the work.

Initarray() is passed the number of items in the list, the pointer to the first element of the array and finally the list of values to be put into the array. In the case of integers, the actual values are put on the stack. With pointers to char's, the compiler puts the string somewhere (we don't really care where) and puts the pointer to it on the stack.

Arrays of pointers to strings are nice, because the strings can be of varying length, but the array elements are always the same size, just two bytes.

That's it for this month. Next time I hope to have a review of the latest introl compiler which now includes floats and longs. I will also start going over some of the C source code for the standard functions that come with run time packages. If you have the introl, intersoft, Telecon or Microware compiler, then you have them in C already. Dugger and Word's Worth supply them in assembler. I don't know about SWTP's or TSC's packages.

I hope that by doing this, we can kill two birds with one stone by studying some more C code in detail and by learning the guts of some of the functions at the same time. Till then.

```
/*
 * date.c   rev: 1
 * n f comno
 *
 * last edit:   1/15/83
 *
 * This program, given the date, returns the day of
 * the week.  It was adapted from a version given in
 * "The C Programming Language" by Kernighan and Ritchie.
 * The algorythm was lifted intact, with the array
 * initializers redone for Small C compilers.
 *
 * Program is normally given a date.  If no date is given,
 * then the program will fetch the date from FLEX.
 *
 */


#define TRUE 1
#define FALSE 0
#define DAYS 7
#define MONTHS 13   /* need a "fudge" */

int noleap[MONTHS];
int leap[MONTHS];
char *daynames[DAYS];
char *monthnames[MONTHS];

main(argc, argv)
    int argc;
    char *argv[];
    {
    int mm,dd,yy;
    char *p;

    /* initialize the arrays */
    init_array(MONTHS,noleap,0,31,28,31,30,31,30,31,31,30,31,30,31);
    init_array(MONTHS,leap,0,31,29,31,30,31,30,31,31,30,31,30,31);
    init_array(MONTHS,monthnames,"","Jan","Feb","Mar","Apr","May","Jun",
                                 "Jul","Aug","Sep","Oct","Nov","Dec");
    init_array(DAYS,dayname,"Sun","Mon","Tue","Wed","Thu","Fri","Sat");

    if (argc < 2)
        {
        printf("\nUsage:  date mm,dd,yyyy\n");
        exit();
        }

    p = argv[1];
    mm = atoi(p);
    while(*p++ != ',') ;
    dd = atoi(p);
    while(*p++ != ',') ;
    yy = atoi(p);
    date(mm,dd,yy);
    }


/*
 * calculate the day of the week and
 * print it to the terminal.
 *
 */
date(month,day,year)
    int month, day, year;
    {
    int *daysin;
    int sd,sm,sy;  /* (s)aved values for printing */

    /* save the originals */
    sd = day;
    sm = month;
    sy = year;
    if (isleap(year))
        daysin = leap;
    else
        daysin = noleap;

    /* check the month */
    if (month < 1  ||  month > 12)
        {
        printf("\nmonth out of range\n");
        return;
        }

    /* check the day of the month */
    if (day < 1  ||  day > daysin[month])
        {
        printf("\nday of month out of range\n");
        return;
        }

    /* get day of the year */
    while (month > 1)
        day += daysin[--month];

    /* now... make the day (mod 7) offset from Jan. 1 0000 */
    if (year > 0)
        {
        --year;
        day += year;
        /* leap year correction */
        day += year/4 - year/100 + year/400;
        }

     /* Jan 1 0000 was a Sunday... all done */
     printf("%s,  %s,%d,%d\n",daynames[day%7],monthnames[sm],sd,sy);
    }


isleap(year)
    int year;
    {
    if (year % 4 != 0)
        return(FALSE);
    if (year % 100 != 0)
```

```
        return(TRUE);
    if (year % 400 != 0)
        return(FALSE);
    return(TRUE);
    }


/*
 * Initialize an array of int's or
 * pointers to char's
 */
init_array(arscnt,array_add,list)
    int arscnt;
    int *array_add;
    int list;           /* this is used only to set first address */
    {
    int i, *arscnt;

    arscnt = &list;     /* point to the start of the list */

    for (i = 0; i < arscnt; i++)
        *array_add++ = *arscnt++;
    }

atoi(s)
    char *s;
    {
    int n;

    n = 0;
    while(isdigit(*s))
        n = (10 * n) + (*s++ - '0');
    return(n);
    }
```

```
                *
                * C.TXT          Rev: 1
                *
                * M F Conno
                *
                * Last edit:    1/9/83
                *
                *
                * This program aids in the production of Ward's Worth
                * C programs.  It runs in the utility area and invokes
                * CPP, CC, COPT, ASMB and RLOAD.  Prior to invoking
                * each pass, it asks the user whether or not to go on.
                * If the user answers yes, then the output file for
                * that phase is deleted.
                *
                * The program is invoked: "C FILESPEC".
                *
                * The source file may come from any disk, but all others
                * will be created on the working drive.  The source file
                * extension will default to ".C", but you may state it
                * explicitly without problems.  All other files will use
                * the default extensions recommended by Ward's Worth in
                * the Version 2 manual.
                *
                * This program is placed in the Public Domain and may
                * be freely copied and used without compensation to,
                * or acknowledgement of the author.
                *
                *
                * Flex equates
                *
C080  FLXBUF  equ   $C080
C100  UTILAREA equ  $C100
CB40  SYSFCB  equ   $CB40
CC0C  WRKDRV  equ   $CC0C
CC14  FLXPNT  equ   $CC14
CD15  GETCHR  equ   $CD15
CD03  WARMS   equ   $CD03
CD1E  PSTRING equ   $CD1E
CD24  PCRLF   equ   $CD24
CD2D  GETFIL  equ   $CD2D
CD4B  LOADMAD equ   $CD4B
0406  FMS    equ   $0406
                *
                * constants
                *
0004  EOL     equ   $04
000C  DELCOD  equ   $0C
000D  CR      equ   $0D
0020  SPACE   equ   $20
0020  BITS    equ   $20
                *
                * local storage
                *
C100            org   UTILAREA
C100            rmb   20
C114  new_st    equ   *
C114  delfls    rmb   1         delete file flag
                *
                * simulated FCB for filespec storage
                *
                *
C115  pad       rmb   3
C118  drive     rmb   1
C119  name      rmb   8
C121  ext       rmb   3
                *
```

```
                * strings
                *
                *
                * NOTE -- extension strings MUST be
                *         EXACTLY 3 characters long
                *
C124 43 50 50 00  cpp     fcc   "CPP",0
C129 43 00 00     cppex1  fcc   "C",0,0
C12D 43 50 50     cppex2  fcc   "CPP"
                *
C12E 43 43 00     cc      fcc   "CC",0
C131 41 53 4D     cex     fcc   "ASM"
                *
C134 43 4F 50 54  copt    fcc   "COPT",0
C138 00
C139 54 58 54     coptex  fcc   "TXT"
                *
C13C 41 53 4D 42  asmb    fcc   "ASMB",0
C140 00
C141 52 45 4C     asmbex  fcc   "REL"
C144 2B 53 4C 59  asmbops fcc   "+SLY",0
C148 00
                *
C149 52 4C 4F 41  rload   fcc   "RLOAD",0
C14D 44 00
C14F 4C 4E 4B     rloadex fcc   "LNK"
                *
C152 44 6F 20 79  prompt  fcc   "Do you want to continue? ",4
C156 6F 75 20 77
C15A 61 6E 74 20
C15E 74 6F 20 63
C162 6F 6E 74 69
C166 6E 75 65 3F
C16A 20 04
                *
                *
                * The main program starts here
                *
                * register usage
                *
                *    x - utility
                *    y - FLEX command line pointer
                *    u - utility
                *
                *
C16C 32   8C A5   start   leas  new_st,PCR
C16F 30   8C A3           leax  pad,PCR   set the file name into storage
C172 BD   CD2D            jsr   GETFIL
C175 30   8C A1           leax  name,PCR
C178 6D   84              tst   ,x
C17A 1027 0162            lbeq  exit      we need a filename!
                *
                * preprocess -- the input is special, and
                *               the extension and drive are
                *               handled differently
                *
C17E 108E C080            ldy   #FLXBUF
C182 30   8C 9F           leax  cpp,PCR
C185 17   00BB            lbsr  ldcmd
C188 17   00C3            lbsr  ldspace
C18B 30   8C 8A           leax  drive,PCR input drive may not be default
C18E A6   84              lda   ,x+
C190 8B   30              adda  #'0
C192 A7   A0              sta   ,y+
C194 86   2E              lda   #'.
C196 A7   A0              sta   ,y+
C198 30   8C B6           leax  ext,PCR   test for non- efault extension
C19B 6D   84              tst   ,x
C19D 26   06              bne   lab1
C19F 30   8C B6           leax  cppex1,PCR
C1A2 17   00AE            lbsr  setext    set source file into buffer
C1A5 17   00B9    lab1    lbsr  ldname
C1A8 17   00A3            lbsr  ldspace
C1AB 30   8D FF7C         leax  cppex2,PCR set output spec into buffer
C1AF 17   00A1            lbsr  setext
C1B2 17   00AC            lbsr  ldname
C1B5 7F   C114            clr   delfls    set flag to "delete"
C1B8 17   00E8            lbsr  idos
                *
                * compile
                *
C1BB 30   8D FF4F         leax  cc,PCR
C1BF 17   0083            lbsr  ldcmd
C1C2 17   0089            lbsr  ldspace
C1C5 17   0099            lbsr  ldname
C1C8 17   0083            lbsr  ldspace
C1CB 30   8D FF62         leax  cex,PCR
C1CF 17   0091            lbsr  setext
C1D2 17   008C            lbsr  ldname
C1D5 7F   C114            clr   delfls
C1D8 17   00CB            lbsr  idos
                *
                * optimize
                *
C1DB 30   8D FF55         leax  copt,PCR
C1DF 17   0063            lbsr  ldcmd
C1E2 17   0069            lbsr  ldspace
C1E5 17   0079            lbsr  ldname
C1E8 17   0063            lbsr  1 space
C1EB 30   8D FF4A         leax  coptex,PCR
C1EF 17   0061            lbsr  setext
C1F2 17   006C            lbsr  ldname
C1F5 7F   C114            clr   delfls
C1F8 17   00AB            lbsr  idos
                *
                * assemble
                *
```

```
                        *
C1FB 30  8D FF3D        leax   asmb,PCR
C1FF 17  0043           lbsr   ldcmd
C202 17  0049           lbsr   ldspace
C205 17  0059           lbsr   ldname
C208 17  0043           lbsr   ldspace
C20B 30  8D FF32        leax   asmbex,PCR
C20F 17  0041           lbsr   setext
C212 17  004C           lbsr   ldname

C215 17  0036           lbsr   ldspace
C218 30  8D FF28        leax   asmbaps,PCR
C21C 17  0026           lbsr   ldcmd
C21F 7F  C114           clr    delflg    redundant here...
C222 17  0081           lbsr   jdos
                        *
                        * link
                        *
C225 30  8D FF20        leax   rload,PCR
C229 17  0019           lbsr   ldcmd
C22C 17  001F           lbsr   ldspace
C22F 30  8D FF1C        leax   rloadex,PCR
C233 17  001D           lbsr   setext
C236 17  0028           lbsr   ldname
C239 7F  C114           clr    delflg
C23C 7C  C114           inc    delflg
C23F 17  0064           lbsr   jdos
C242 16  0098           lbra   exit
                        *
                        *
                        *
                        * ldcmd -- load a command string into the buffer
                        *
                        * entry:  x - points to command string
                        *         y - points to FLEX cmd line
                        *
                        * exit:   x,y - altered
                        *
C245 A6  80             ldcmd  lda    x+
C247 27  04             beq    ldcmd1
C249 A7  A0             sta    y+
C24B 20  F8             bra    ldcmd
C24D 39                 ldcmd1 rts
                        *
                        *
                        *
                        * ldspace -- load a space into the buffer
                        *
                        * entry:  y - points to FLEX command buffer
                        *
                        * exit:   a - altered
                        *         y - altered
                        *
C24E 86  20             ldspace lda    #SPACE
C250 A7  A0                     sta    y+
C252 39                 rts
                        *
                        *
                        *
                        * setext -- set the extension into the file
                        *           storage area
                        *
                        * entry:  x - points to extension
                        *
                        * exit:   a,b,x,u altered
                        *
C253 C6  03             setext ldb    #3
C255 33  8D FEC8        leau   ext,PCR
C259 A6  80             set1   lda    x+
C25B A7  C0                    sta    u+

C25D 5A                 decb
C25E 26  F9             bne    set1
C260 39                 rts
                        *
                        *
                        *
                        * ldname -- load file name and ext into the buffer
                        *
                        * entry:  y - points into FLEX cmd line
                        *
                        * exit:   a,b,x,y - altered
                        *

C261 30  8D FEB4 ldname leax   name,PCR  point to name
C265 C6  08             ldb    #8       name is 8 char's ma
C267 A6  80             ldnm1  lda    x+
C269 27  05             beq    ldnm2    NULL means end of name
C26B A7  A0             sta    y+
C26D 5A                 decb
C26E 26  F7             bne    ldnm1
C270 86  2E             ldnm2  lda    #'.      add the period
C272 A7  A0             sta    y+
C274 C6  03             ldb    #3       ext is 3 char's max
C276 30  8D FEA7        leax   ext,PCR  load in the extension
C27A A6  80             ldnm3  lda    x+
C27C 27  05             beq    ldnm4    NULL means end of ext
C27E A7  A0             sta    y+
C280 5A                 decb
C281 26  F7             bne    ldnm3
C283 39                 ldnm4  rts
                        *
                        *
```

```
                        *
                        * delfil -- delete a file, all errors are
                        *           ignored, we don't care if it
                        *           wasn't there to be deleted
                        *
                        *
                        * exit:   a,b,x,u - altered
                        *
C284 8E  CB40   delfil  ldx    #SYSFCB  point to system FCB
C287 33  8D FEBD        leau   drive,PCR and to local filespec storage
C28B B6  CC0C           lda    WRKDRV   set drive to working
C28E A7  C4             sta    u
C290 34  10             pshs   x        save FCB pointer
C292 30  03             leax   3,x      move pointer to FCB filespec
C294 C6  0C             ldb    #12      set filespec size
C296 A6  C0     delf1   lda    u+       move filespec into FCB
C298 A7  80             sta    x+
C29A 5A                 decb
C29B 26  F9             bne    delf1
C29D 35  10             puls   x        recover FCB pointer
C29F 86  0C             lda    #DELCOD  set delete code
C2A1 A7  84             sta    x
C2A3 7E  D406           jmp    FMS      execute and return
                        *
                        *
                        *
                        * jdos -- set up for, and jsr to FLEX
                        *
                        * entry:  y - points into FLEX cmd line
                        *
                        *
                        * exit:   y - set to start of FLEX cmd line
                        *         a,b,x,u - altered
                        *
C2A6 8E  C080   jdos    ldx    #FLXBUF  set start of buffer
C2A9 34  30             pshs   x,y      save both
C2AB 86  04             lda    #EOL     terminate line for printing
C2AD A7  A4             sta    ,y
C2AF 8D  CD24           jsr    PCRLF
C2B2 8D  CD1E           jsr    PSTRING  show the user what we're up to!
C2B5 30  8D FE99        leax   prompt,PCR prompt for action
C2B9 8D  CD1E           jsr    PSTRING
C2BC 8D  CD15           jsr    GETCHR   get his answer
C2BF 8A  20             ora    #BBIT5   set to lower case
C2C1 81  6E             cmpa   #'n
C2C3 27  1B             beq    exit
C2C5 8D  CD24           jsr    PCRLF
C2C8 7D  C114           tst    delflg   deleting output file?
C2CB 26  03             bne    jdos1
C2CD 17  FFB4           lbsr   delfil
C2D0 35  30     jdos1   puls   x,y
C2D2 86  0D             lda    #CR      terminate line for DOS
C2D4 A7  A4             sta    y
C2D6 34  10             pshs   x        save pointer to start of line
C2D8 BF  CC14           stx    FLXPNT   reset FLEX's line pointer
C2DB 8D  CD4B           jsr    OUTCRLF
C2DE 35  A0             puls   y,PC     restore y to line pointer
                        *
                        *
                        *
C2E0 7E  CD03   exit    jmp    WARMS
                        end    start

0 ERROR(S) DETECTED
```

FLEXIBLE P SYS

FLEX BASED FLEXIBLE PRINT SYSTEM
By Kenneth Drexler
311 Wilson Way
Larkspur, CA 94939

If you are a long-time user of Technical Systems Consultants' FLEX operating system, you may have become dissatisfied like I did with the format limitations of FLEX's file printing utilities. I wrote two new utilities to solve the problem: PRINT.SYS and SETPRINT.CMD. These utilities work together to provide a flexible printing system for FLEX and my printer.

PRINT.SYS is similar to the one which TSC provides in the FLEX System Manual - that is, it is loaded by P.CMD and sends characters to your printer. However, it does much more. The PRINT.SYS below can print a variable number of lines per page, print single, double or more spaced pages, print a left margin of a size you choose and send transparent codes to your printer to control it.

SETPRINT.CMD allows you to display and change the print format used by PRINT.SYS either before or after printing starts. When it is run, SETPRINT stops the Print Spooler and loads PRINT.SYS, if necessary. The program then displays the current PRINT.SYS format. The information displayed is (1) Left Margin, (2) Line Spacing, (3) Page Length, (4) Printer Line Length and (5) Printer Address. It then asks if you want to change any of these items. If you do, it asks for the changed information.

Before returning to FLEX, PRINT.SYS adjusts the line length in FLEX to compensate for the left margin which you set and displays the Printer Control Codes used for controlling the printer. As a convenience, the program accepts both upper and lower case data.

The Printer Control Code function in PRINT.SYS is modeled after the system used by Alford & Associates in its SCREDITOR III word processing system. It allows up to 20 Printer Control Codes of the form, "(Up Arrow)DD", to be imbedded in the text of a document. These codes instruct PRINT.SYS to send a code to the printer to cause it to perform a predefined function such as Turn On Italics, Start Underline, End Underline, etc. When PRINT.SYS encounters these codes in a document, it sends the corresponding code to the printer. The Printer Control Code itself is not sent to the printer. The control codes are, thus, transparent to the printer. The printer code corresponding to each Printer Control Code must be specifically selected for the printer used with the system.

Program Description

PRINT.SYS consists of three routines: PINIT, PCHK and POUT. The first two are similar to those shown in the FLEX manuals and are loaded into the space provided in FLEX at $ACC0 (or $CCC0).

The third, POUT, is greatly different. It appears at lines 95 to 97 and 108 to 276 in the listing below and is the heart of PRINT. SYS. It must be loaded outside of FLEX because it is too large to fit in the space provided in FLEX. It contains the code which adds flexibility to TSC's PRINT.SYS. It also contains the RAM locations in which the current left margin, line spacing, page length, printer line length and printer address data are stored. When PRINT.SYS is loaded from disk these locations are set to the default values shown in the code below. These default settings allow you to use FLEX's P.CMD without first running SETPRINT.

The core of POUT is at lines 124 to 143. This portion of the program examines each character received in the A Register to determine what to do with it. To assist in the decision, several flags are used. The most important are the CCFLAG, which shows that an "Up Arrow" has been received and is still being processed, and the CRFLAG, which shows that the previous character was a "Carriage Return". This portion of the program branches to the routines which handle a Printer Control Code, Carriage Return, Line Feed and Form Feed. Any other character is sent to the printer without further processing.

PRINT.SYS must be placed in RAM where it will not be erased by the loading of programs such as BASIC or the text files for EDIT. (It MUST be in RAM.) In effect this means that it must be protected by being placed in memory isolated from the memory running from $0000 up or by adjusting MEMEND in FLEX to protect the program. My system has 1K of RAM at $F000 which I use.

SETPRINT is similar to other utilities in FLEX and loads into the FLEX utility space. It is a transient and does not need any special memory protection.

The code below is designed for an Epson 80F/T printer with GRAPHTRAX, not GRAPHTRAX PLUS. If you have a different printer you may need to change the printer codes to match your printer. For example, the Epson 80 with GRAPHTRAX does not implement an automatic underline function. For this reason, the Start and End Underline Codes are not used in my system. If your printer provides an underline function, the printer codes will have to be changed.

Both SETPRINT.CMD and PRINT.SYS are written for my non-standard Monitor in which the address of the printer is stored in RAM so that it can be changed without reprogramming a ROM. Both programs load the Printer address from memory before using it in the program. (It is this address which SETPRINT sets in response to the "Printer Address?" question.) If you have a fixed address for your printer, the printer output routines can be modified to load a fixed address. The "Printer Address?" question should then be eliminated from SETPRINT.

My printer drivers work with a PIA-driven parallel printer. If you have a serial printer, substitute your own printer driver for POUT20 and your own printer initialize program for PTRINZ.

In the program below, there are only 20 printer codes and each is limited to two bytes of information. These limitations reflect the control code structure of my Epson 80F/T with GRAPHTRAX. If your printer needs longer or more control codes, the program can easily be modified to accommodate your needs.

That is it. I hope the program is of use. - - -

```
                        **** SET PRINT COMMAND ****
                        ****************************************************
                        *
                        *       SET PRINT COMMAND
                        *
                        * DATE: OCTOBER 9, 1982
                        *
                        * COMMAND FORMAT: "SETPRINT"
                        *
                        * THIS PROGRAM LOADS "PRINT.SYS" AND DISPLAYS
                        * OR CHANGES THE PRINT FORMAT.  THE FORMAT
                        * VARIABLES SUPPORTED ARE: LEFT MARGIN,
                        * LINE SPACING AND PAGE LENGTH.
                        *
                        * "PRINT.SYS" ALLOWS PRINTER CONTROL CODES
                        * TO BE IMBEDDED IN TEXT AND BE TRANSMITTED
                        * TO THE PRINTER.  EACH CODE IS PRECEEDED
                        * BY AN "UP ARROW".  THE CODES CAN RANGE
                        * FROM 0 TO 19, EXCLUDING 7 AND 13.
                        *
                        * AFTER LOADING, THE PROGRAM LOADS "PRINT.SYS"
                        * FROM THE DISK IF IT IS NOT PRESENT.  IT THEN
                        * DISPLAYS THE CURRENT PRINTER INFORMATION AND
                        * ASKS IF ANY CHANGES ARE WANTED.  IF SO,
                        * THE PROGRAM GETS NEW DATA AND DISPLAYS IT.
                        * SETPRINT THEN DISPLAYS THE PRINT CODE
                        * ASSIGNMENTS AND THEIR MEANINGS.
                        *
                        * ON EXIT THE LINE LENGTH IN FLEX IS ADJUSTED
                        * TO LIMIT THE PRINTED LINE TO THE LENGTH OF
                        * THE PRINTER LINE LENGTH MINUS 2.
                        *
                        * THE COMMAND DISPLAYS AND CHANGES THE
                        * ADDRESS OF THE PRINTER PIA.
                        *
                        * WHILE THE PROGRAM IS RUNNING, PRINTING IS
                        * SUSPENDED, ALLOWING THE PRINT INFORMATION
                        * TO BE CHANGED DURING PRINTING.
                        *
                        ****************************************************
                        *
                        * SYSTEM EQUATE
A000                    FLEX    EQU     $A000     FLEX 2.0
                        *FLEX EQU $C000 FLEX 9

                        *FLEX EQUATES
ACE4                    POUT    EQU     F EX+$0CE4
A840                    FCB     EQU     FLEX+$0840
B406                    OUTS    EQU     FLEX+$1406
AD30                    LOAD    EQU     FLEX+$0D30
AD2D                    GETFIL  EQU     FLEX+$0D2D
AD33                    SETEXT  EQU     FLEX+$0D33
AD27                    NXTCH   EQU     FLEX+$0D27
AD03                    WARMS   EQU     FLEX+$0D03
AD1E                    PSTRNG  EQU     FLEX+$0D1E
AD3F                    RPTERR  EQU     FLEX+$0D3F
ACC0                    PINIT   EQU     FLEX+$0CC0
```

```
AD15            GETCHR  EQU   FLEX+$0D15
AD18            INBUF   EQU   FLEX+$0D18
AD46            INDEC   EQU   FLEX+$0046
AD45            OUTAUX  EQU   FLEX+$0045
AD24            PCRLF   EQU   FLEX+$0024
AD39            OUTDEC  EQU   FLEX+$0039
AD18            PUTCHR  EQU   FLEX+$0D18
AD42            GETHEX  EQU   FLEX+$0D42
AC04            TTYWD   EQU   FLEX+$0C04  TTYSET LINE LENGTH
ACFC            PRBUSY  EQU   FLEX+$0CFC  PRINTER BUSY FLAG

                *PRINT.SYS EQUATES
F260            JPOUT   EQU   $F260
F280            SPCNT   EQU   JPOUT+0     LEFT MARGIN SPACES
F281            LSPACE  EQU   JPOUT+1     LINE SPACING
F282            PAGLN   EQU   JPOUT+2     PAGE LENGTH
F283            PLINE   EQU   JPOUT+3     PRINTER LINE LENGTH
F032            PTRADR  EQU   $F032       PRINTER ADDRESS

A100                    ORG   FLEX+$0100
A100 20 06      SETPR   BRA   SETP1

A102 01         VER     FCB   1           VERSION

                *VARIABLES
A103            PRFLAG  RMB   1
A104            TEMP    RMB   2
A106            MSGPNT  RMB   2

A108 B6 AC FC   SETP1   LDA A PRBUSY      STOP PRINT.CMD
A10B B7 A1 03           STA A PRFLAG
A10E 7F AC FC           CLR   PRBUSY
A111 CE A2 46   SETP9   LDX   #MSG1       PRINT HEADER
A114 BD AD 1E           JSR   PSTRNG
A117 B6 AC E4           LDA A PDUT        "PRINT.SYS" LOADED?
A11A 81 39              CMP A #$39
A11C 26 1E              BNE   SETP3       YES, SKIP LOAD
A11E CE AB 40           LDX   #FCB        NO, LOAD IT
A121 86 01              LDA A #1          OPEN FILE FOR READ
A123 A7 00              STA A 0,X
A125 BD B4 06           JSR   FMS
A128 27 08              BEQ   SETP2       NO ERROR
A12A A6 01              LDA A 1,X         HANDLE ERRORS
A12C 81 04              CMP A #4
A12E 26 24              BNE   REPERR
A130 CE A2 5A           LDX   #MSG2       PRINT "NOT FOUND"
A133 20 27              BRA   EXIT1

A135 86 FF      SETP2   LDA A #$FF        SET BINARY READ
A137 A7 31              STA A $31,X
A139 BD AB 30           JSR   LOAD

                *COMMAND LOOP
A13C BD 2A      SETP3   BSR   RSTAT       REPORT STATUS, CHANGE?
A13E BD AB 18           JSR   INBUF       GET ANSWER
A141 BD A8 27           JSR   NXTCH
A144 84 5F              AND A #$5F        FORCE UPPER CASE
A146 81 59              CMP A #'Y
A148 27 79              BEQ   SSTAT       SET STATUS
A14A 81 4E              CMP A #'N
A14C 27 08              BEQ   EXIT
A14E CE A2 70           LDX   #MSG3       PRINT "WHAT"
A151 BD AB 1E           JSR   PSTRNG
A154 20 E6              BRA   SETP3

A156 BD AD 3F   REPERR  JSR   RPTERR      REPORT ERROR
A159 CE A3 22   EXIT    LDX   #MSG8       PRINT PTR CTRL CODES
A15C BD AD 1E   EXIT1   JSR   PSTRNG
A15F B6 A1 03           LDA A PRFLAG      TURN PRINTING ON
A162 B7 AC FC           STA A PRBUSY
A165 7E AD 03           JMP   WARMS

                *PRINT STATUS
A168 CE A2 77   RSTAT   LDX   #MSG4       PRINT CAPTION
A16B BD AD 1E           JSR   PSTRNG
A16E CE A2 8B           LDX   #MSG5       PRINT ID LINES
A171 8D 2C              BSR   PMSG1       SET POINTER
A173 CE F2 80           LDX   #SPCNT      PRINT LEFT MARGIN
A176 8D 37              BSR   DECOUT
A178 8D 22              BSR   PMSG
A17A CE F2 81           LDX   #LSPACE     PRINT LINE SPACING
A17D 8D 30              BSR   DECOUT
A17F 8D A1 9C           JSR   PMSG
A182 CE F2 82           LDX   #PAGLN      PRINT PAGE LENGTH
A185 8D 28              BSR   DECOUT
A187 8D 13              BSR   PMSG
A189 CE F2 83           LDX   #PLINE      PRINT PTR LINE LEN.
A18C BD A1 AF           JSR   DECOUT
A18F 8D 0B              BSR   PMSG
A191 CE F0 32           LDX   #PTRADR     PRINT PRINTER ADDR.
A194 BD AD 45           JSR   OUTADR
A197 8D 27              BSR   CRLF
A199 81 01              BSR   PMSG        ASK "CHANGE?"
A19B 39                 RTS

                *PRINT MESSAGES
A19C FE A1 06   PMSG    LDX   MSGPNT      RESTORE X
A19F A6 00      PMSG1   LDA A 0,X         GET DATA
A1A1 08                 INX
A1A2 81 04              CMP A #$04        END?
A1A4 27 05              BEQ   PMSG2
A1A6 BD AD 18           JSR   PUTCHR
A1A9 20 F4              BRA   PMSG1

A1AB FF A1 06   PMSG2   STX   MSGPNT      SAVE X
A1AE 39                 RTS

                *PRINT TWO DECIMAL DIGITS
A1AF A6 00      DECOUT  LDA A 0,X         GET DATA AT X
A1B1 B7 A1 05           STA A TEMP+1
A1B4 CE A1 04           LDX   #TEMP       POINT AT WORKSPACE
A1B7 6F 00              CLR   0,X         CLEAR IT
A1B9 5F                 CLR B             SUPPRESS ZEROS
A1BA BD AD 39           JSR   OUTDEC
A1BD 8D 01              BSR   CRLF        PRINT LINE
A1BF 39                 RTS

                *BRANCH BOOSTER
A1C0 7E AD 24   CRLF    JMP   PCRLF

                *SET STATUS ROUTINE
A1C3 CE A2 8B   SSTAT   LDX   #MSG5       POINT AT ID LIST
A1C6 8D D7              BSR   PMSG1       PRINT AND SET POINTER
A1C8 BD A2 24           JSR   BUFIN       GET DATA
A1CB B7 F2 80           STA A SPCNT       SET MARGIN
A1CE 8D F0              BSR   CRLF
A1D0 8D CA      SSTAT1  BSR   PMSG
A1D2 8D 50              BSR   BUFIN       GET DATA
A1D4 81 05              CMP A #$05        CHECK SPACING
A1D6 25 0E              BLO   SSTAT2
A1D8 CE A2 FD           LDX   #MSG6       PRINT "ENTER 1..."
A1DB BD AD 1E           JSR   PSTRNG
A1DE CE A2 9B           LDX   #MSGCSA     RESET POINTER
A1E1 FF A1 06           STX   MSGPNT
A1E4 20 EA              BRA   SSTAT1      LINE SPACING AGAIN

A1E6 B7 F2 81   SSTAT2  STA A LSPACE      SET LINE SPACING
A1E9 8D D5              BSR   CRLF
A1EB 8D AF              BSR   PMSG
A1ED 8D 35              BSR   BUFIN       GET DATA
A1EF B7 F2 82           STA A PAGLN       SET PAGE LENGTH
A1F2 8D CC              BSR   CRLF
A1F4 8D A6              BSR   PMSG
A1F6 8D 2C              BSR   BUFIN       GET DATA
A1F8 B7 F2 83           STA A PLINE       SET PRINTER LINE
A1FB B0 F2 80           SUB A SPCNT       SET FLEX LINE
A1FE 4A                 DEC A
A1FF 4A                 DEC A             MINUS 2
A200 B7 AC 04           STA A TTYWD
A203 8D B8              BSR   CRLF
A205 8D 95              BSR   PMSG        ASK FOR PRINTER ADDR.
A207 86 3F              LDA A #'?
A209 BD AD 18           JSR   PUTCHR
A20C 86 20              LDA A #$20
A20E BD A8 18           JSR   PUTCHR
A211 BD AD 18           JSR   INBUF
A214 BD AE 42           JSR   GETHEX      READ ADDRESS
A217 25 27              BCS   NONUM       ERROR EXIT
A219 5D                 TST B
A21A 27 24              BEQ   NONUM
A21C FF F0 32           STX   PTRADR
A21F 8D 9F              BSR   CRLF
A221 7E A1 59           JMP   EXIT

A224 86 3F      BUFIN   LDA A #'?         PRINT QUES.
A226 BD AD 18           JSR   PUTCHR
A229 86 20              LDA A #$20
A22B BD AD 18           JSR   PUTCHR
A22E BD AD 18           JSR   INBUF       GET DATA
A231 BD AD 46           JSR   INDEC
A234 25 0A              BCS   NONUM       ERROR
A236 5D                 TST B
A237 27 07              BEQ   NONUM
A239 FF A1 06           STX   TEMP
A23C CE A3 12   NONUM   LDX   #MSG7       PRINT "INVALID NUM"
A23F 7E A1 5C           JMP   EXIT1

A242 53         MSG1    FCC   /SET PRINT COMMAND/
A253 0D                 FCB   $D,$A,4
A256 04

A257 22         MSG2    FCC   /"PRINT.SYS" NOT FOUND/
A26C 04                 FCB   4

A26D 57         MSG3    FCC   /WHAT??/
A273 04                 FCB   4

A274 50         MSG4    FCC   /PRINT INFORMATION/
A285 00                 FCB   $0,$A,4

A286 0D         MSG5    FCB   $D,$A
A288 4C                 FCC   /LEFT MARGIN: /
A296 04                 FCB   4
A297 4C         MSG5A   FCC   /LINE SPACING: /
A2A6 04                 FCB   4
A2A7 50                 FCC   /PAGE LENGTH: /
A2B4 04                 FCB   4
A2B5 50                 FCC   /PRINTER LINE LENGTH: /
A2C9 04                 FCB   4
A2CA 50                 FCC   /PRINTER ADDRESS $: /
A2DE 04                 FCB   4
A2DF 0D                 FCB   $D,$A
A2E2 43                 FCC   /CHANGE FORMAT (Y OR N)? /
A2FC 04                 FCB   4

A2FD 45         MSG6    FCC   /ENTER 1, 2, 3 OR 4/
A30F 0D                 FCB   $D,$A,4

A312 49         MSG7    FCC   /INVALID NUMBER /
A321 04                 FCB   4

A322 20         MSG8    FCC   / - CONTROL CODE        00 - PRINTER BACKSPACE/
A350 0D                 FCB   $D,$A
A353 30                 FCC   /01 - START UNDERLINE   02 - END UNDERLINE/
A37E 0D                 FCB   $D,$A
A380 30                 FCC   /03 - START EMPHASISED  04 - END EMPHASISED/
A3AC 0D                 FCB   $D,$A
```

```
A3AE 30              FCC   /05 - START DBL. WIDTH    06 - END DBL. WIDTH/
A3B0 0D              FCB   $D,$A
A3B1 30              FCC   /07 - BELL               08 - NOT USED/
A402 0D              FCB   $D,$A
A404 30              FCC   /09 - NOT USED           10 - LINE FEED/
A428 0D              FCB   $D,$A
A42D 31              FCC   /11 - HEAD HOME LEFT     12 - FORM FEED/
A454 0D              FCB   $D,$A
A456 31              FCC   /13 - CARRIAGE RETURN /
A46E 0D              FCB   $D,$A
A46D 31              FCC   /14 - ITALICS ON         15 - ITALICS OFF/
A490 0D              FCB   $D,$A
A498 31              FCC   /16 - DBL. STRIKE ON     17 - DBL. STRIKE OFF/
A4C5 0D              FCB   $D,$A
A4C7 31              FCC   /18 - COMPRESSED ON      19 - COMPRESSED OFF/
A4F3 0D              FCB   $D,$A,4

            *SET FCB FILE NAME
A843                 ORG   FLEX+$0843
A843 FF              FCB   $FF
A844 50              FCC   /PRINT/
A849 00              FCB   0,0,0
A84C 53              FCC   /SYS/

                     END   SETPR

NO ERROR(S) DETECTED
```

```
  1                  ****REVISED PRINT.SYS PROGRAM****
  2                  **********************************************
  3                  *
  4                  * DATE: NOVEMBER 1, 1982
  5                  *
  6                  * FILE NAME: PRINTSYS.TXT
  7                  *
  8                  * SEE FLEX USER'S MANUAL, P. 3.7
  9                  * DESIGNED FOR AN EPSON 80 F/T PRINTER WITH
 10                  * GRAFTRAX DRIVEN BY A PIA USING A
 11                  * CENTRONICS-TYPE HANDSHAKE.
 12                  *
 13                  * PRINTS A VARIABLE NUMBER OF SPACES ON THE
 14                  * LEFT MARGIN, NUMBER OF LINES PER PAGE AND
 15                  * LINE FEEDS PER CARRIAGE RETURN.  THE LATTER
 16                  * ALLOWS DOUBLE, TRIPLE OR MORE SPACING.
 17                  * ALL VARIABLES ARE OUTSIDE THE OPERATING
 18                  * CODE AND ARE AVAILABLE TO BE SET BY THE
 19                  * MONITOR OR A FORMATTING PROGRAM
 20                  * SUCH AS "SETPRINT".
 21                  *
 22                  * RECEIPT OF FORM FEED CLEARS THE PROGRAM'S
 23                  * LINE COUNT.
 24                  *
 25                  * THE PROGRAM ALSO ACCEPTS AND PROCESSES
 26                  * PRINTER CONTROL CODES OF THE FORM:
 27                  * "UP ARROW DD" WHICH ARE EMBEDDED IN TEXT.
 28                  * "DD" CAN RANGE FROM 0 TO 19.  UPON RECEIPT
 29                  * OF A CONTROL CODE, THE CORRESPONDING
 30                  * CODE TO CAUSE THE INDICATED ACTION
 31                  * IS SENT TO THE PRINTER.
 32                  * INVALID AND OUT OF RANGE CODES ARE IGNORED.
 33                  * THE CONTROL CODES ARE NOT PRINTED.
 34                  *
 35                  * RECEIPT OF "UPARROW 12" (PRINTER FORM FEED)
 36                  * CLEARS THE PROGRAM'S LINE COUNT.
 37                  *
 38                  **********************************************
 39                  *
 40                  * DEFAULT DATA
 41  0000            SPCD    EQU   0          LEFT MARGIN
 42  0001            LSP     EQU   1          LINE SPACING
 43  003A            PAGL    EQU   58         PAGE LENGTH
 44  0050            PTRWD   EQU   80         PRINTER LINE LENGTH
 45
 46                  * SYSTEM EQUATE
 47  A000            FLEX    EQU   $A000      FOR FLEX 2.0
 48                  * FLEX EQU $C000 FOR FLEX 9
 49
 50                  * FLEX EQUATE
 51  AD36            ADDBX   EQU   FLEX+$0D36
 52
 53                  * EQUATES
 54  000C            FFEED   EQU   $0C
 55  F032            PTRADR  EQU   $F032      MONITOR PRINTER ADDR.
 56
 57                  *** VERSION NUMBER CODE
 58                  * THIS CODE ALLOWS VERSION.CMD TO READ
 59                  * THE CORRECT VERSION NUMBER
 60
 61  F300                    ORG   $F300      OVERWRITTEN LATER
 62  F300 00 00              FDB   0          WASTE SPACE
 63  F302 01                 FCB   1          VERSION NUMBER
 64
 65                  *** PRINTER INITIALIZATION
 66                  * SETS D0 - D7 TO OUTPUTS, C1 TO POSITIVE
 67                  * TRANSITION AND C2 TO FOLLOW CR3.  ALL
 68                  * INTERRUPTS ARE DISABLED.
 69
 70  ACC0                    ORG   FLEX+$0CC0
 71  ACC0 FE F0 32  PINIT    LDX   PTRADR     GET PRINTER ADDR.
 72  ACC3 6F 01              CLR   1,X        ADDRESS DDR
 73  ACC5 86 FF              LDA A #$FF       SET D0-D7 AS OUTPUT
 74  ACC7 A7 00              STA A 0,X
 75  ACC9 86 3E              LDA A #$3E       SET STATUS
 76  ACCB A7 01              STA A 1,X
 77  ACCD 4F                 CLR A
 78  ACCE BD F2 B2           JSR   POUT20     OUTPUT DATA
 79  ACD1 7F F2 86           CLR   LINCT      CLEAR FLAGS
 80  ACD4 7F F2 87           CLR   CRFLAG
 81  ACD7 20 14              BRA   PINIT1     BRANCH AROUND PCHK
 82
 83                  *** CHECK FOR PRINTER READY
 84                  * RETURN WITH "MINUS" IF READY
 85                  * SAVES A, B AND X REGISTERS
 86
 87  ACD9                    ORG   FLEX+$0CD9
 88  ACD9 FF F2 B4  PCHK     STX   XTEMP      SAVE X
 89  ACDC 36                 PSH A            SAVE DATA
 90  ACDD FE F0 32           LDX   PTRADR     GET PRINTER ADDRESS
 91  ACE0 6D 01              TST   1,X        SET C (READY-B7=1)
 92  ACE2 07                 TPA              SAVE IN A
 93  ACE2 20 03              BRA   PCHK1      BRANCH OVER OUTPUT
 94
 95                  *** PRINTER OUTPUT ROUTINE
 96  ACE4                    ORG   FLEX+$0CE4
 97  ACE4 7E F2 B0  POUT     JMP   JPOUT
 98
 99  ACE7 FE F2 B4  PCHK1    LDX   XTEMP      RESTORE X
100  ACEA 06                 TAP              RESTORE C AND A
101  ACEB 32                 PUL A
102  ACEC 39                 RTS
103
104  ACED 7F F2 B6  PINIT1   CLR   CCFLAG
105  ACF0 7F F2 B9           CLR   ONEDON
106  ACF3 39                 RTS


108  F2B0                    ORG   $F2B0
109                  *** PRINT ROUTINE
110
111                  * VARIABLES
112  F2B0 00         SPCNT   FCB   SPCD       SPACES-LEFT MARGIN
113  F2B1 01         LSPACE  FCB   LSP        LINE FEEDS PER CR
114  F2B2 3A         PAGLN   FCB   PAGL       LINES PER PAGE
115  F2B3 50         PLINE   FCB   PTRWD      PRINTER LINE WIDTH
116  F2B4            XTEMP   RMB   2
117  F2B6            LINCT   RMB   1          LINE COUNT
118  F2B7            CRFLAG  RMB   1          1= PREV. CHAR. WAS CR
119  F2B8            CCFLAG  RMB   1          1= PREV. WAS UPARROW
120  F2B9            ONEDON  RMB   1          1= TENS DIGIT DONE
121  F2BA            CODE    RMB   1          PRINT CONTROL CODE
122  F2BB            TEMP    RMB   2
123
124                  ** MAIN LINE CODE
125  F2BD FF F2 B4  JPOUT    STX   XTEMP      SAVE REGISTERS
126  F2C0 36                 PSH A
127  F2C1 37                 PSH B
128  F2C2 FE F0 32           LDX   PTRADR     GET PRINTER ADDR.
129  F2C5 7D F2 B8           TST   CCFLAG     PREV. CONTROL CODE?
130  F2C8 26 75              BNE   PRCODE     YES, HANDLE IT
131  F2CA 81 0D              CMP A #$0D       CARRIAGE RETURN?
132  F2CC 27 4B              BEQ   CR         YES. HANDLE IT
133  F2CE 81 0A              CMP A #$0A       LINE FEED?
134  F2D0 27 34              BEQ   LFD        YES, HANDLE IT
135  F2D2 81 5E              CMP A #$5E       UPARROW?
136  F2D4 27 29              BEQ   PCODE      YES, HANDLE IT
137  F2D6 81 0C              CMP A #$FFEED    FORM FEED?
138  F2D8 27 1E              BEQ   FORM       YES. HANDLE IT
139  F2DA BD 06              BSR   POUT20     OUTPUT DATA
140  F2DC FE F2 B4  EXIT     LDX   XTEMP      RESTORE REGISTERS
141  F2DF 33                 PUL B
142  F2E0 32                 PUL A
143  F2E1 39                 RTS
144
145                  * OUTPUT DATA (DESTROYS X)
146  F2E2 FE F0 32  POUT20   LDX   PTRADR     GET PRINTER ADDR.
147  F2E5 6D 01     POUT21   TST   1,X        PRINTER READY?
148  F2E7 2A FC              BPL   POUT21
149  F2E9 A7 00              STA A 0,X        PUT DATA
150  F2EB 36                 PSH A            SAVE DATA
151  F2EC 86 36              LDA A #$36       PULSE C2 ON PIA
152  F2EE A7 01              STA A 1,X
153  F2F0 86 3E              LDA A #$3E
154  F2F2 A7 01              STA A 1,X
155  F2F4 A6 00              LDA A 0,X        CLEAR PIA IRQ FLAG
156  F2F6 32                 PUL A            RESTORE A
157  F2F7 39                 RTS
158
159                  ** HANDLE CR, LF, FF, PRINT CODES
160
161                  * HANDLE FORM FEED
162  F2F8 7F F2 B6  FORM     CLR   LINCT      CLEAR LINE COUNT
163  F2FB BD E5              BSR   POUT20     PRINT FORM FEED
164  F2FD 20 DD              BRA   EXIT
165
166                  * SET CONTROL CODE FLAG
167  F2FF 86 FF      PCODE   LDA A #$FF       SET CONTROL CODE FLAG
168  F301 B7 F2 B8           STA A CCFLAG
169  F304 20 D6              BRA   EXIT
170
171                  * HANDLE LINE FEED
172  F306 7D F2 B7  LFD      TST   CRFLAG     LF FOLLOWING CR?
173  F309 27 05              BEQ   LFD1       NO, MIDLINE
174  F30B 7F F2 B7           CLR   CRFLAG     YES. CLEAR FLAG
175  F30E 20 CC              BRA   EXIT
176
177                  * HANDLE MID LINE LINE FEED
178  F310 BD D0      LFD1    BSR   POUT20     PRINT IT
179  F312 7C F2 B6           INC   LINCT      ONE MORE LINE
180  F315 BD 73              BSR   PAGE       PAGE END?
181  F317 20 C3              BRA   EXIT
182
```

```
                         * HANDLE CARRIAGE RETURN
F2E9 8D C7      CR       BSR    POUT20     PRINT CR
F2EB 86 FF               LDA A  #$FF       SET CR FLAG
F2ED B7 F2 87            STA A  CRFLAG

                         * SPACE BETWEEN LINES
F2F0 F6 F2 81            LDA B  LSPACE     GET LINE SPACE COUNT
F2F3 27 0C               BEQ    CR2        NO LINE SPACE???
F2F5 86 0A               LDA A  #$0A       PRINT LINE FEED
F2F7 8D B9      CR1      BSR    POUT20
F2F9 7C F2 86            INC    LINCT      LINE COUNT+1
F2FC 5A                  DEC B             DECREMENT COUNT
F2FD 26 F8               BNE    CR1
F2FF 8D 59               BSR    PAGE       PAGE END?

                         * INSERT MARGIN SPACES
F301 86 20      CR2      LDA A  #$20       LOAD SPACE
F303 F6 F2 80            LDA B  SPCNT      LOAD SPACE COUNT
F306 27 05               BEQ    CR4        IF 0, SKIP
F308 8D A8      CR3      BSR    POUT20     SPACE LOOP
F30A 5A                  DEC B
F30B 26 FB               BNE    CR3        DONE? NO, LOOP
F30D 20 9D      CR4      BRA    EXIT

                         * PROCESS PRINTER CONTROL CODES
F30F 81 30      PRCODE   CMP A  #$30       RANGE CHECK 0 - 9?
F311 25 3F               BLO    PRCOD4
F313 81 3A               CMP A  #$3A
F315 24 3B               BHS    PRCOD4
F317 84 0F               AND A  #$0F       GET RIGHT NIBBLE
F319 7D F2 89            TST    ONEDON     FIRST DIGIT DONE?
F31C 26 12               BNE    PRCOD2     YES
F31E 81 01               CMP A  #$01       LESS THAN 1?
F320 22 30               BHI    PRCOD4     >1, ERROR
F322 26 02               BNE    PRCOD1     = ZERO
F324 86 0A               LDA A  #$0A       1 = TEN
F326 B7 F2 8A   PRCOD1   STA A  CODE       SAVE FIRST DIGIT
F329 86 FF               LDA A  #$FF       SET ONE DONE FLAG
F32B B7 F2 89            STA A  ONEDON

                         * READ SECOND CONTROL DIGIT, SEND CODE
F330 BB F2 8A   PRCOD2   ADD A  CODE       COMBINE WITH 10'S
F333 81 0C               CMP A  #12        FORM FEED CODE?
F335 26 03               BNE    PRCOD3     NO
F337 7F F2 86            CLR    LINCT      YES, CLEAR LINE COUNT
F33A 48         PRCOD3   ASL A             CODE TIMES TWO
F33B 16                  TAB               MOVE TO B
F33C CE F3 6D            LDX    #PCTBL     POINT AT CODE TABLE
F33F BD AD 36            JSR    ADDBX      GET ADDR OF PTR CODE
F342 A6 00               LDA A  0,X        GET FIRST BYTE
F344 FF F2 8B            STX    TEMP       SAVE POINTER
F347 BD F2 B2            JSR    POUT20     PRINT IT
F34A FE F2 8B            LDX    TEMP       RESTORE X
F34D A6 01               LDA A  1,X        GET SECOND BYTE
F34F BD F2 B2            JSR    POUT20     PRINT IT
F352 7F F2 88   PRCOD4   CLR    CCFLAG     CLEAR CODE FLAG
F355 7F F2 89            CLR    ONEDON     CLEAR ONE DONE FLAG
F358 20 B3               BRA    CR4        EXIT

                         * CHECK PAGE LENGTH AND HANDLE
F35A B6 F2 82   PAGE     LDA A  PAGLN      LOAD PAGE LENGTH
F35D 27 0D               BEQ    PAGE1      IF ZERO, NO CHECK
F35F B1 F2 86            CMP A  LINCT      PAGE DONE?
F362 24 08               BHS    PAGE1      NO, SKIP
F364 7F F2 86            CLR    LINCT      YES, CLEAR COUNT
F367 86 0C               LDA A  #FFEED     SEND FORM FEED
F369 BD F2 B2            JSR    POUT20     OUTPUT IT
F36C 39         PAGE1    RTS

                         * PRINTER CODE TABLE
                         * DESIGNED FOR AN EPSON 80 F/T
                         * WITH GRAPHTRAX

F36D 7F 00      PCTBL    FDB    $7F00      00 PRINTER BACKSPACE
F36F 00 00               FDB    $0000      01 START UNDERLINE
F371 00 00               FDB    $0000      02 END UNDERLINE
F373 1B 45               FDB    $1B45      03 START EMPHASISED
F375 1B 46               FDB    $1B46      04 END EMPHASISED
F377 1B 53               FDB    $1B53      05 START DBL. WIDTH
F379 1B 54               FDB    $1B54      06 END DBL. WIDTH
F37B 07 00               FDB    $0700      07 BELL
F37D 00 00               FDB    $0000      08 NOT USED
F37F 00 00               FDB    $0000      09 NOT USED
F381 0A 00               FDB    $0A00      10 LINE FEED
F383 1B 3C               FDB    $1B3C      11 HOME HEAD
F385 0C 00               FDB    $0C00      12 FORM FEED
F387 0D 00               FDB    $0D00      13 CARRIAGE RETURN
F389 1B 34               FDB    $1B34      14 START ITALICS
F38B 1B 35               FDB    $1B35      15 END ITALICS
F38D 1B 47               FDB    $1B47      16 START DBL. STRIKE
F38F 1B 48               FDB    $1B48      17 END DBL. STRIKE
F391 1B 50               FDB    $1B50      18 START COMPRESSED
F393 1B 51               FDB    $1B51      19 END COMPRESSED

                         END

NO ERROR(S) DETECTED
```

MODEM

GENERAL.

The program BELLMAN is meant for external communication via FLEX between two modems. All transmission to modem and keyboard is made via ACIA 6850. When starting up BELLMAN, there are a number of FLEX-vectors that are changed but when exiting BELLMAN nicely (via ctrl F) they are restored.

BELLMAN is written in assembly language for 6809 and FLEX-9, version 2:8.1. Users who have other versions should check the patch addresses. The program is based on interrupts, therefore both ACIA's have to be wired for IRQ.

The benefits with IRQ's are, that no switches have to be installed for Rx/Tx and that you have full control of the system. The program has three different interrupt levels:

1. character in from modem
2. character in from keyboard
3. end of program (carrier loss)

The carrier IRQ starts automatically program execution via a modem, when someone calls your computer (Bellman on line). Executable programs in BELLMAN are all those that do not conflict with BELLMAN in memory and use FLEX I/O routines.

LIMITATIONS.

Programs using interrupts may not be executed, as they change BELLMAN's IRQ-vectors. Programs that, when exiting to FLEX WARMS, restores FLEX I/O or IRQ vectors, may not be executed (e.g EXEC.CMD). Programs that use real-time may not execute correctly. The easiest way to get rid of the problems mentioned, is to create a BELLMAN-systemdisk, that only has executable programs.

HARDWARE REQUIREMENTS.

Computersystem with 6809, FLEX ver.2:8.1. Serial interface via 6850 ACIA to modem and keyboard and both wired for IRQ. If automatic answering facilities are needed, make the following changes to the modem-ACIA (if you have a MP-S):

1. Disconnect 6850 pin 23 (DCD) from gnd.
2. Deliver 1489 pin 1 and 3.
3. Connect 6850 pin 23 to 1489 pin 3 (if you don't need reader c)
4. Connect 1489 pin 1 to telephone modem RS-232 Cannon pin 6, (DCD out).

If you don't have MP-S, you must notice that ACIA needs TTL-levels and that IRQ is generated when DCD goes high.

GETTING STARTED.

Syntax : BELLMAN,BELLFILE,<+>

where BELLFILE is a TXT-file that is transmitted when BELLMAN is called. The file could be a 'Welcome to Bellman', system-info, etc. If option + is used, BELLMAN assumes that there is a program called BELLCOM.CMD on the system drive. This program is always called whenever BELLMAN is restarted. BELLCOM could have a time/date routine or a login-procedure. Be sure to rename whatever program you use to BELLCOM.CMD.

MISCELLANEOUS.

BELLMAN also has an internal FLEX-command, WRITE. This command is used whenever one wants to save a file on disk. This mode always has to be ended with ctrl D which is the only possibility to reenter BELLMAN.

So, have a good time in installing an automatic host computer at your home, connected to your telephone line.

Yours truly

Esko Antikainen (SM5AKP)
Arvodesv 17
S-126 46 Hagersten
SWEDEN

Avo Kask (SMOKVO)
Hoderv 8
S-184 02 Osterskar
SWEDEN


```
************************
* Welcome to BELLMAN *
************************

Available is FLEX-DOS with its utility programs.
System Drive=0 and Work Drive=1.
NB !! BELLMAN does NOT echo.
CTRL C ---> Restart the program from the beginning
CTRL D ---> The last character in WRITE-command
CTRL E ---> ESC pause/continue. CR return to prompt
WRITE  ---> Open file for writing
CAT    ---> Catalog. ( CAT 0.XX lists all files on drive 0 starting with XX )
LIST   ---> Reads a textfile ( LIST ZZ reads the fileZZ.TXT from drive 1 )

Only filename executes a file with extension .CMD ( eg. CAT )

The connection is broken when carrier is missing.
```

```
                * BELLMAN
                * PROGRAM FOR FLEX ACCESS WITH
                * EXTERNAL IN- AND OUTPUT VIA MODEM
                *
                * WRITTEN BY ESKO ANTIKAINEN ( SM5AKP )
                * AND       AVO KASK        ( SMOKVO )

```

```
                        * IN OCTOBER 1982
                        *
                        * SYNTAX 'BELLMAN,BELLFILE,<+>'
                        * + OPTION READS FILE 'BELLCOM.CMD' FROM SYSTEM DRIVE
                        * THIS FILE IS CALLED AT EVERY START (RESTART) AND
                        * MAY BE FOR EXAMPLE A TIME AND DATE TO TERMINAL.
                        * OR LOGIN-PROCEDURE

                        * EQUATES

              E008  MODPOR  EQU   $E008     MODEMPORT
              E004  KEYPOR  EQU   $E004     KEYBOARDPORT
              DFCB  IRQVEC  EQU   $DFCB     IRQVECTOR
              CB40  FCB     EQU   $CB40     SYSTEM FCB

CC12                        ORG   USRTBL
CC12 B26C                   FDB   WRITE    SET WRITE ROUTINE TO FLEX

B000                        ORS   $B000    PROGRAMSTART

B000 20  14       STARTA    BRA   STARTS
B002 01                     FCB   1        VERSION NUMBER

                        * CONTROL CHRS IN BELLMAN

B003 18             PAUS    FCB   $18      ESC FOR PAUS
B004 04             FILEST  FCB   4        CTRL D FOR WRITE FILEMD
B005 03             RESTA   FCB   3        CTRL C FOR RESTART
B006 06             RETFLE  FCB   6        CTRL F FOR FLEX RETURN
B007 08             BACKSP  FCB   8        BACKSPACE CHARACTER

                        * SCRATCH AREA

B008                STAFLG  RMB   1        TRANSMIT INFOTEXT ?
B009                STAVEC  RMB   2
B00B 00             PAUSFL  FCB   0
B00C                MEME    RMB   2
B00E                FILEMD  RMB   2
B010                PATCH   RMB   1
B011                MONPAT  RMB   3
B014                FLEPAU  RMB   1
B015 FF             UFLG    FCB   $FF       EXECUTE COMMAND FILE IF 0

                        * ROUTINE FOR READING BELLFILE

B016 BE  CC2B     STARTB    LDX   MEMEMD
B019 BF  B00C               STX   MEME     SAVE MEMEMD
B01C BE  B000               LDX   #STARTA
B01F 30  1F                 LEAX  -1,X
B021 BF  CC2B               STX   MEMEMD   NEW MEMEMD
B024 B6  CC09               LDA   PAUSE
B027 B7  B014               STA   FLEPAU   SAVE SYSTEMPAUSE
B02A 7F  CC09               CLR   PAUSE    NO PAUSE IN FLEX
B02D BE  CB40               LDX   #FCB     SYSFCB
B030 108E B31F              LDY   #STATXT  STARTADDRESS BELLFILE
B034 BD  CD2D               JSR   GETFIL
B037 25  62                 BCS   ERR1
B039 B6  01                 LDA   #1
B03B BD  CD33               JSR   SETEXT   SET DEF .TXT
B03E B6  01                 LDA   #1
B040 A7  84                 STA   ,X
B042 BD  D406               JSR   FMS      OPEN FILE
B045 26  54                 BNE   ERR1

B047 BD  D406     LOOP      JSR   FMS      SET CHAR
B04A 26  4F                 BNE   ERR1
B04C 81  0D                 CMPA  #$0D     END OF LINE ?
B04E 26  04                 BNE   LOOP1
B050 A7  A0                 STA   ,Y+
B052 86  0A                 LDA   #$0A

B054 A7  A0       LOOP1     STA   ,Y+
B056 20  EF                 BRA   LOOP

B058 86  04       FILOK     LDA   #4
B05A A7  A4                 STA   ,Y       END OF FILE CHAA
B05C BE  CB40               LDX   #FCB
B05F A7  84                 STA   ,X
B061 BD  D406               JSR   FMS      CLOSE FILE
B064 26  35                 BNE   ERR1
B066 BE  CC14               LDX   #CC14    SET BUFFER POINTER
B069 A6  84                 LDA   ,X
B06B 81  2B                 CMPA  #'+
B06D 26  03                 BNE   ENERR
B06F 7F  B015               CLR   UFLG     GOING TO EXECUTE COMMAND FILE

                        * ENABLE ERROR INFO TO MODEM

B072 B6  CDE4     ENERR     LDA   #CDE4    FLEX2B1 I/O PATCH
B075 B7  B016               STA   PATCH
B078 B6  39                 LDA   #$39
B07A B7  CDE4               STA   #CDE4

                        *DISABLE 'MON' COMMAND

B07D BE  D370               LDX   #$D370
B080 10BE B011              LDY   #MONPAT
B084 A6  80                 LDA   ,X+
B086 A7  A0                 STA   ,Y+
B088 EC  84                 LDD   ,X
B08A ED  A4                 STD   ,Y
B08C 8E  D370               LDX   #$D370
B08F B6  7E                 LDA   #$7E
B091 A7  80                 STA   ,X+
B093 CC  CD03               LDD   #$CD03
B096 A7  84                 STA   ,X
>B098 7E  B086              JMP   SEGNY    CONTINUE IN PROG

B09B A6  01       ERR1      LDA   1,X      ERROR BYTE
B09D B1  08                 CMPA  #8       READ PAST END OF FILE ?
B09F 27  07                 BEQ   FILOK
B0A1 BD  CD3F               JSR   APTERR
B0A4 B9  D403               JSR   FMSCLS
B0A7 8E  B00C               LDX   #MEME
B0AA BF  CC2B               STX   MEMEMD
B0AD B6  B014               LDA   FLEPAU
B0B0 B7  CC09               STA   PAUSE    RESTORE SYSTEMPAUSE
B0B3 7E  CD03               JMP   WARMS

B0B6 10FF B009    SEGNY     STS   STAVEC    SAVE SYSTEMSTACK

B0BA 10FE B009    SEGA      LDS   STAVEC    RESTORE STACK

B0BE BE  E008     SEGTM     LDX   #MODPOR   PORTADRESS MODEM
B0C1 86  03                 LDA   #3
B0C3 A7  84                 STA   ,X       MASTER RESET
B0C5 86  95                 LDA   #%10010101 ENABLE INTERRUPT
B0C7 A7  04                 STA   ,X       SETUP KEYBOARDACIA
B0C9 BE  E004               LDX   #KEYPOR
B0CC A7  84                 STA   ,X       SETUP MODEMACIA
B0CE 7F  B008               CLR   STAFLG   SEND STARTTEXT

B0D1 10FE B009    START     LDS   STAVEC   RESTORE STACK
B0D5 BD  D403               JSR   FMSCLS   CLOSE ALL FILES
B0D8 BE  B113               LDX   #MODIN
B0DB BF  CD0A               STX   INCH+1   NEW INPUTROUTINE
B0DE BE  B116               LDX   #MODOUT
B0E1 BF  CD10               STX   OUTCH+1  NEW OUTPUTROUTINE
B0E4 BE  B12E               LDX   #$1RQ
B0E7 BF  DFCB               STX   IRQVEC   NEW IRQ ADDRESS
B0EA 7F  CC21               CLR   $CC21    FLEX I/O FLAG
B0ED 7F  CC22               CLR   $CC22    ONT SWITCH
B0F0 7F  CC23               CLR   $CC23    INPUT SWITCH
B0F3 7D  B00B               TST   STAFLG
B0F6 26  0A                 BNE   START2
B0F8 8E  B21A               LDX   #OKTXT
B0FB BD  CD1E               JSR   PSTRNG
B0FE 3C  EF                 CWAI  #%11101111 WAIT FOR START
B100 20  CF                 BRA   START    READY FOR COMMAND

B102 BE  B20E     START2    LDX   #XMEMY
B105 BD  CD1E               JSR   PSTRNG
B108 8E  CB40               LDX   #$CB40   SYSTEM FCB
B10B BD  CB1B               JSR   INBUFF
B10E BD  CD4D               JSR   DOCMND   EXECUTE THE COMMAND
B111 20  BE                 BRA   START

                        * OUT/IN MODEM

B113 3C  EF       MODIN     CWAI  #%11101111 ENABLE IRQ
B115 39                     RTS
```

```

0116 73    CC22     MODOUT  COM    OUTSW     OUT TO TERMINAL
0119 BD    CD18             JSR    PUTCHR
011C 7F    CC22             CLR    OUTSW     OUT TO MODEM
011F 34    02               PSHS   A

0121 B6    E000     MODEM   LDA    MODPOR
0124 84    02               ANDA   #2
0126 27    F9               BEQ    MODEM     NOT READY YET
0128 35    02               PULS   A
012A B7    E009             STA    MODPOR+1  SEND THE CHAR
012D 39                     RTS

                    * IRQ ROUTINE IS CALLED WHEN CHAR IN

012E 7D    0008     IRQ     TST    STAFLG    TRANSMIT STARTTEXT ?
0131 26    12               BNE    IRQ1
0133 8E    031F             LDX    #STATXT
0136 BD    CD1E             JSR    PSTRNG    TRANSMIT STARTTXT
0139 86    FF               LDA    #$FF
013B B7    0008             STA    STAFLG    STARTTEXT TRANSMITTED
013E BD    02ED             JSR    DOCOM
0141 BD    01CE             JSR    CLRIRQ
0144 3B                     RTI

0145 BD    01FA     IRQ1    JSR    MODIRQ
0148 24    11               BCC    NXTTST
014A 81    0005             CMPA   RESTA     RESTART ?
014D 27    5E               BEQ    RETUR
014F A7    61               STA    1,S       SAVE CHARACTER IN ACCA FOR RTI
0151 73    CC22             COM    OUTSW
0154 BD    CD18             JSR    PUTCHR    ECHO TO TERMINAL
0157 7F    CC22             CLR    OUTSW
015A 3B                     RTI

>015B BD   01B6     NXTTST  JSR    KEYIRQ
015E 24    4D               BCC    RETUR     MUST BE CARRIER
0160 81    0006             CMPA   RETFLE    FLEX RETURN ?
0163 27    0E               BEQ    STOP
0165 81    0005             CMPA   RESTA     RESTART ?
0168 27    43               BEQ    RETUR
>016A BD   0116             JSR    MODOUT    ECHO AND TRANSMIT CHAR
016D A7    61               STA    1,S       SAVE CHAR IN ACCA FOR RTI
016F 3B                     RTI

0170 BD    CD3F     ERROR   JSR    RPTERR

0173 BD    D403     STOP    JSR    FMSCLS    CLOSE ALL FILES
0176 86    03               LDA    #3
0178 B7    E000             STA    MODPOR    CLOSE MODEMPORT
017B B7    E004             STA    KEYPOR
017E 86    15               LDA    #$15
0180 B7    E004             STA    KEYPOR
0183 BE    000C             LDX    MEME
0186 BF    CC2B             STX    MEMEND    RESTORE SYSTEM MEMORY
0189 B6    0014             LDA    FLEPAU
018C B7    CC09             STA    PAUSE     RESTORE SYSTEMPAUSE
018F 7F    CC12             CLR    USRTBL
0192 7F    CC13             CLR    USRTBL+1  NO EXTERN USERTABLE
0195 B6    0010             LDA    PATCH
0198 B7    CDE4             STA    $CDE4     RESTORE FLEX I/O
019B 8E    0011             LDX    #MONPAT
019E 108E  D370             LDY    #$D370
01A2 A6    80               LDA    ,X+
01A4 A7    A0               STA    ,Y+
01A6 EC    84               LDD    ,X
01A8 ED    A4               STD    ,Y        RESTORE 'MON' COMMAND
01AA 7E    CD03             JMP    WARMS

01AD 8E    0EBA     RETUR   LDX    #0EBA
01B0 AF    6A               STX    10,S      NEW RETURN ADDRESS
>01B2 BD   01CE             JSR    CLRIRQ
01B5 3B                     RTI

                    * SUBROUTINES FOR POLLING IRQ
                    * IF ORQ CALL THEN CARRY IS SET
                    * ELSE CARRY IS CLEARED

01B6 8E    E004     KEYIRQ  LDX    #KEYPOR
01B9 A6    84               LDA    ,X        STATUS REG
01BB 2A    4E               BPL    NOIRQ

01BD A6    01       JAIRQ   LDA    1,X       CHARACTER IN
01BF 84    7F               ANDA   #$7F      MASK PARITY
01C1 1C    EF               ANDCC  #%11101111 ENABLE IRQ
01C3 81    0003             CMPA   PAUS      DO PAUSE/CONTINUE
01C5 27    15               BEQ    SWITCH
01C8 81    0D               CMPA   #$D       CR FOR RETURN
01CA 27    1B               BEQ    CRTST
01CC 20    29               BRA    CONT

                    * CLEARS IRQFLAGS

01CE 8E    E000     CLRIRQ  LDX    #MODPOR
01D1 A6    84               LDA    ,X
01D3 A6    01               LDA    1,X
01D5 8E    E004             LDX    #KEYPOR
01D8 A6    84               LDA    ,X
01DA A6    01               LDA    1,X
01DC 39                     RTS

                    * PAUSROUTINE WHEN 'ESC' AND RETURN WHEN 'CR'

01DD 73    0008     SWITCH  COM    PAUSFL

01E0 7D    0008     TESTA   TST    PAUSFL
01E3 26    FB               BNE    TESTA
01E5 20    10               BRA    CONT

01E7 7D    0008     CRTST   TST    PAUSFL
01EA 27    0B               BEQ    CONT
01EC 7F    0000             CLR    PAUSFLG   NO PAUSE
01EF 32    62               LEAS   2,S       STACK BEFORE SUBROUTINE
01F1 8E    CC16             LDX    ESCRET    RETURNADDRESS
01F4 AF    6A               STX    10,S      NEW RETURN FROM IRQ
01F6 3B                     RTI

01F7 1A    01       CONT    SEC
01F9 39                     RTS

01FA 8E    E000     MODIRQ  LDX    #MODPOR
01FD A6    84               LDA    ,X        STATUS REG
01FF 2A    0A               BPL    NOIRQ
0201 85    04               BITA   #%00000100
0203 27    B8               BEQ    JAIRQ
0205 B6    0005             LDA    RESTA
0208 1A    01               SEC              RETURN CHR IN A
020A 39                     RTS

020B 1C    FE       NOIRQ   CLC
020D 39                     RTS


                    * TEXT STRINGS

020E 42 65 6C 6C    MENY    FCC    /Bellman >> /,4
0212 6D 61 6E 20
0216 3E 3E 20 04
021A 42 65 6C 6C    OKTXT   FCC    /Bellman on line /,4
021E 6D 61 6E 20
0222 6F 6E 20 6C
0226 69 6E 65 20
022A 04
022B 46 69 6C 65    TBLTXT  FCC    /Filename ? /,4
022F 6E 61 6D 65
0233 20 3F 20 04
0237 57 61 69 74    WAITXT  FCC    /Wait, saving file on disk/,4
023B 2C 20 73 61
023F 76 69 6E 67
0243 20 66 69 6C
0247 65 20 6F 6E
024B 20 64 69 73
024F 6B 04
0251 46 69 6E 69    FILTXT  FCC    /Finish writing with CTRL D/,4
0255 73 68 20 77
0259 72 69 74 69
025D 6E 67 20 77
0261 69 74 68 20
0265 43 54 52 4C
0269 20 44 04
```

```
                        * ROUTINE FOR WRITE COMMAND
B26C 57 52 49 54    WRITE   FCC     /WRITE/
     0270 45
0271 00                     FCB     $
0272 B275                   FDB     WRITE1
0274 00                     FCB     $           END OF TABLE

B275 8E  0220       WRITE1  LDX     #TBLTIT
0278 BD  CD1E               JSR     PSTRNG
027B 8E  CB40               LDX     #FCB
027E BD  CD18               JSR     INBUFF
0281 BD  CD2D               JSR     GETFIL
0284 1025 0092              LBCS    FILERR
0288 B6  01                 LDA     #1          EXT= .TXT
028A BD  CD33               JSR     SETEXT
028D B6  02                 LDA     #2
028F A7  84                 STA     ,X
0291 BD  D406               JSR     FMS         OPEN FILE
0294 1026 00B2              LBNE    FILERR
0298 108E 0000              LDY     #$0000      POINT MEMORY START
029C 34  10                 PSHS    X
029E 8E  0251               LDX     #FILTXT
02A1 BD  CD1E               JSR     PSTRNG
02A4 BD  CD24               JSR     PCRLF
02A7 35  10                 PULS    X

02A9 3C  EF         WRILOP  CWAI    #$EF        WAIT FOR CHR IN
02AB B1  0007               CMPA    BACKSP
02AE 26  04                 BNE     LABI
02B0 31  3F                 LEAY    -1,Y
02B2 20  F5                 BRA     WRILOP

02B4 A7  A0         LABI    STA     ,Y+         PUT CHR IN MEMORY
02B6 B1  0004               CMPA    FILEST      FILE END ?
02B9 27  02                 BEQ     WRIRED
02BB 20  EC                 BRA     WRILOP

02BD 4F             WRIRED  CLRA
02BE A7  84                 STA     ,X
02C0 10BF B00E              STY     FILEND
02C4 108E 0000              LDY     #$0000
02C8 34  10                 PSHS    X
02CA 8E  B237               LDX     #WAITXT
02CD BD  CD1E               JSR     PSTRNG
02D0 35  10                 PULS    X

02D2 10BC B00E      DISLOP  CMPY    FILEND
02D6 27  06                 BEQ     FILCLO
02D8 A6  A0                 LDA     ,Y+         GET SAVED CHR
02DA B0  D406               JSR     FMS         WRITE TO FILE
02DD 1026 0039              LBNE    FILERR

02E0 7D  B015       DOCOM   TST     UFLG
02E3 27  01                 BEQ     DOC00
02E5 39                     RTS                 NO COMMAND FILE

02E6 8E  C800       DOC00   LDX     #$C800      LINEBUFFER
02E9 108E B30E              LDY     #CONTXT

02ED A6  80         DOC0    LDA     ,X+
02EF A7  80                 STA     ,X+
02F1 B1  0D                 CMPA    #$D
02F4 27  02                 BEQ     DOC1
02F6 20  F6                 BRA     DOC0

02F8 8E  C800       DOC1    LDX     #$C800
02FB BF  CC14               STX     $CC14
02FE BD  CD4B               JSR     DOCMND
0301 39                     RTS

0302 42 45 4C 4C    CONTXT  FCC     /BELLCOM.CMD/ OPTION (+) FILENAME
     0306 43 4F 4D 2E
     030A 43 4D 44
030D 0D                     FCB     $0D

030E BD  CD3F       FILERR  JSR     RPTERR
0311 20  C4                 BRA     FILCLO

0313 B31F           STATIT  RMB     1

                            END     STARTA

0 ERROR(S) DETECTED

SYMBOL TABLE:

ADDBI  CD36  BACKSP B007  BECHR  CC07  BEGA   00BA  BEGIN  00BE
SEGNY  00B6  BSCHR  CC08  BUFPTR CC14  CMNDRV DE0F  CLASS  CD21
CLRIRQ B1CE  COLDS  C000  COMFLG CC2B  COMREG E01B  COMTXT B30E
CONT   B1F7  CPUTYP CC33  CRTST  B1E7  CURCHR CC1B  CURCOL CC29
DATE   CC0E  DATREG E01B  DELCHR CC01  DEPTH  CC03  DISLOP B2D2
DOC0   B2FA  DOC00  B2F3  DOC1  B304  DOCMND CD4B  DOCOM  B2ED
DRVREG E014  DRVSEL DE0C  EJECT CC0B  ENERR  B072  EOLCHR CC02
ERR1   B09B  ERRNAM CC2D  ERROR B170  ERRTYP CC20  ESCCHR CC0A
ESCRET CC16  FCB    CB40  FIEFLG CC2F  FILELO B2E3  FILEND B00E
FILERR B31A  FILEST B004  FILOK  B05B  FILTXT B251  FINADR CC26
FLEPAU B014  FMS    D406  FMSCLS D405  FOTADR CC24  GETCHR CD15
GETFIL CD2D  GETNEX CD42  INBUFF CD1B  INCH   CD09  INCHR  CD0C
INDEC  CD4B  INIT   DE12  INPSW  CC23  IOFLS  CC21  IRQ    B12E
IRQ1   B145  IRQVEC DFCB  JAIRQ  B1BD  KEYIRQ B1B6  KEYPOR E804
LAB1   B2B4  LOOFFS CC1B  LINEOF C480  LOAD   CD30  LOOP   B647
LOOP1  B054  LSTTRM CC11  MEME   600C  MEMEND CC20  MENY   020E
MODEM  B121  MODIN  B113  MODIRQ B1FA  MODOUT B116  MODPOR E008
MONFAT B011  MOIRQ  B2BB  NULL   CC05  NXTCH  CD27  NXTTST B15B
OKTXT  B21A  OUTADR CD45  OUTCH  CD0F  OUTCH2 CD12  OUTDEC CD39
OUTHEX CD3C  OUTSW  CC22  PATCH  B010  PAUS   B003  PAUSE  CC09
PAUSFL S008  PCRLF  CD24  PDUTCH CCE4  PREADY CC08  PRINIT CCC8
PRVCHR CC19  PSTRNG CD1E  PUTCHR CD1B  RDSEC  0E00  RENTER CD06
RESTA  B005  RESTOR DE09  RETFLE B006  RETUR  B1AD  RPTERR CD3F
RSTRIO CD2A  SECREG E01A  SETEXT CD33  STAFLG B00B  START  B0D1
STARTI B102  STARTA B000  STARTB B016  STATXT B31F  STAVEC B009
STOP   B173  SWITCH B1D0  SYSDRV CC0B  TABCHR CC06  TBLTIT B120
TEST0  B1E0  TRYREG E019  UFLG   B015  USRTBL CC12  VERSEC DE06
```

68000 DUAL PORTS

DUAL-PORTED RAM FOR THE MC68000 MICROPROCESSOR

Prepared by
Trey West
MicroProcessor Applications Engineer
Motorola Microprocessor Division
Austin, Texas

INTRODUCTION

Dual-ported RAM provides a means for multiprocessor systems to exchange data without directly interfering with each other. In most systems this data exchange involves a master MPU passing information to a slave MPU. For example, a host MPU may need to transfer information to a graphic processing circuit to direct the display operation. Redundant processing schemes may require a "checking processor" to compare the results of several MPUs operating simultaneously. Whatever the application, some form of communication between processors is required.

A dual-ported RAM has, as its name implies, two independent ports or address/data/control buses. This scheme simply allows two processors to access the same memory contents without interfering with each other. Thus, depending on the amount of dual-ported RAM that is available, messages, instructions, data, etc. may be transfered from one processor to the other.

Access to the dual-ported RAM is controlled by one or more semaphore registers. A semaphore register is simply a memory location set aside as a flag to indicate whether or not a dual-ported RAM is currently in use. If the semaphore bit is set, one of the two processors is currently using the dual-ported RAM space and the other processor is not allowed access. Other semaphore registers could be defined to indicate messages available, contents changed, etc.

The MC68000 TAS (test and set) instruction supports this semaphore concept. The TAS instruction reads the semaphore register, determines if the MSB of the semaphore register is set, and if it is clear it sets the MSB of the register. If the MSB of the semaphore register is already set, the TAS instruction simply reports (via the condition code register) that the resource

is allocated to another processor at this time. Besides replacing several instructions with one, the TAS instruction executes the entire read/modify/write cycle in one indivisible bus cycle, thus alleviating the possibility of both processors reading the semaphore as clear and both assuming they have the resource after setting the semaphore flag bit (MSB).

The cycle timing considerations for the dual-processor system discussed in this application note were calculated using two MC68000L8 (8-MHz clock) microprocessors. In addition, a separate 12.5 MHz arbitration clock was used to clock the dual-ported RAM arbitration circuit. The same system could be used with any other MC68000 microprocessors and arbitration clock speeds. However, the arbitration clock speed should be sufficiently high to ensure that the cycle timing is commensurate with the system speed.

HARDWARE CONSIDERATIONS

In a multiprocessor system, the operation of each processor may be asynchronous with respect to the other MPUs in the system. With asynchronous operation it is impossible to predict when an MPU will request access to the contents of the dual-ported RAM (hereafter referred to as DPR); therefore, a DPR arbitration circuit is a necessity. As shown in the circuit diagram of Figure 1, four D latches (U3A, U3B, U4A, U4B) form the basis of the arbitration circuit. The first two latches (U3A and U3B) are clocked on opposite phases of a 12.5 MHz arbitration clock. Initially after reset, the state of the four D latches is such that the U4A and U4B $\overline{Q}$ input to OR gates U2A and U2B are low. If one of these two OR gates receives a low chip select input ($\overline{CS1}$, $\overline{CS2}$) it will cause a change in one set of the D latches (U3A, U3B). Since both D latches of the first pair (U3A, U3B) are clocked on opposite phases of the 12.5 MHz arbitration clock, only one will change state, even if the falling edge of both $\overline{CS1}$ and $\overline{CS2}$ signals occur simultaneously. The second set of latches (U4A, U4B) provides a debounce latch for the first set. The debounce latch is required since, if a rising 12.5 MHz arbitration clock edge and the D input both change state at the same time, the corresponding Q output could become unstable for the next 75 nanoseconds. The U4A-U4B set of latches are also clocked by the 12.5 MHz arbitration clock, and approximately 80 nanoseconds later the latched low chip select signal appears at the Q output ($\overline{CS1}$ at U4A, $\overline{CS2}$ at U4B). The latched low chip select signal also presets the other D latch; similarly, the high $\overline{Q}$ output is cross-coupled to the OR gate input of the first D latch. This feedback holds off the access of the other processor until the first processor has finished its access and releases its chip select ($\overline{CS1}$, $\overline{CS2}$) signal.

Once the arbitration circuit has selected the processor which is allowed access to the DPR, the other processor is locked out and cannot gain access to the DPR until the first processor has completed its bus cycle. Locking out the other processor is accomplished by holding off its data transfer acknowledge ($\overline{DTACK}$) signal. Because holding off $\overline{DTACK}$ locks out the other processor, the $\overline{BERR}$ timer signal of the locked out processor should be longer than any $\overline{DTACK}$ delay caused by DPR accesses.

NOTE

The longest $\overline{BERR}$ timeout required would result from a lockout at the start of a TAS instruction. In the case of an 8-MHz MC68000 running with this DPR hardware, the worst case time would be approximately two microseconds. The suggested $\overline{BERR}$ timeout in this case is 10 microseconds.

The latched $\overline{CS1}$ or $\overline{CS2}$ signal (referred to as CSx) is presented to a 74LS164 shift register provided that one or both of the data strobes ($\overline{LDS}$, $\overline{UDS}$) is asserted. Once the signal is present at the serial input of the shift register, it begins propagating through on positive transitions of the 12.5 MHz clock. The shift register outputs become $\overline{CSx}$ Delayed and $\overline{DTACK}$ for either processor. The delay before $\overline{DTACK}$ can be adjusted by using a switch or jumper to tap off the appropriate delayed output of the 74LS164. The cycle ends when the $\overline{CSx}$ signal or both data strobes are negated, clearing the shift register.

The random logic gates attached to the 74LS164s allow the use of the TAS instruction. The TAS instruction simplifies support of the dual port RAM semaphore registers. This instruction is a special case because two bus cycles take place during one $\overline{AS}$ asserted time (see Figure 2). The first bus cycle starts in the same manner as all MC68000 cycles with $\overline{AS}$ and $\overline{LDS}$ or $\overline{UDS}$ asserted (TAS is a byte operation only). At the end of this read cycle the only indication of a complete transfer is the negation of the data strobes. Therefore, negated data strobes must clear the shift register to remove $\overline{CSx}$ Delayed and $\overline{DTACK}$. The next cycle (write) starts by asserting the data strobes ($\overline{LDS}$ or $\overline{UDS}$). An asserted data strobe releases the shift register clear input ($\overline{CSx}$ is also low)and allows application of the $\overline{CSx}$ Latched input to the 74LS164 serial input. The cycle continues as a normal write.

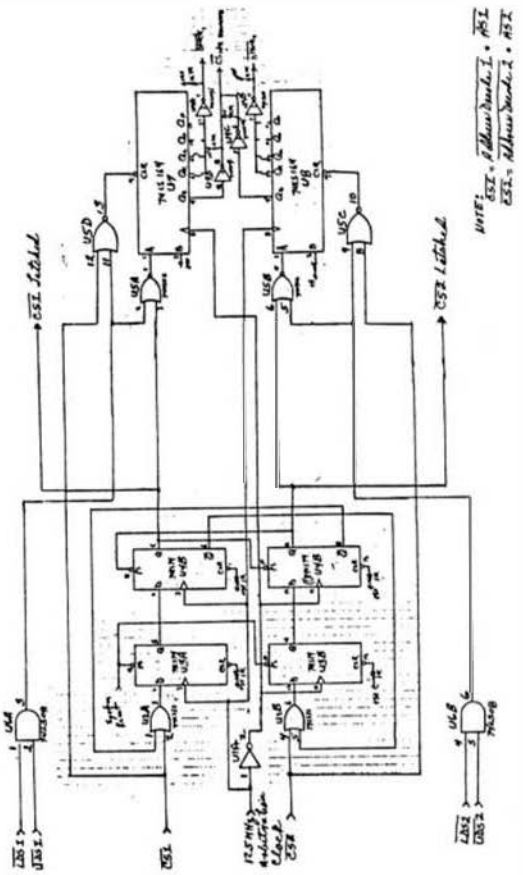


Figure 1. Dual-Ported RAM Bus Arbitration Circuit, Schematic Diagram

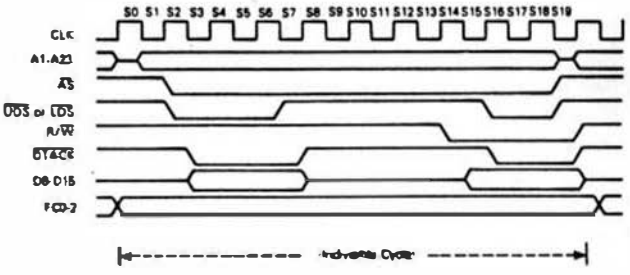


Figure 2. Read-Modify-Write Cycle Timing Diagram

The schematic diagram in Figure 3 shows the necessary MC68B10 static RAMs and buffers required to allow two processors to access the same devices. Octal bi-directional buffers, U10, U11, U16, U17 (74LS245s) provide buffering for the data bus. These devices are controlled by the $\overline{CSx}$ Data input ($\overline{CSx}$ Data signals are the $\overline{CSx}$ Delayed outputs of the 74LS164 shift registers delayed again by passing through a 74LS244). Only one set of data buffers are on at any time as controlled by the arbitration circuitry. Data direction is determined by the state of the corresponding R/$\overline{W}$ line. The address lines are buffered in a similar manner with the $\overline{CSx}$ Latched signal providing the enable input for the two 74LS244s. By using the $\overline{CSx}$ Latched signal (instead of $\overline{CSx}$ Delayed), the address setup time (t_{as}) required by the MC68B10s is met prior to the $\overline{CSx}$ Delayed signal being presented to the chip s[illegible]t inputs of the

MC68B10s (this address setup time is not required by many other memories). A 74LS244 (U13) is used as a multiplexer for the R/$\overline{W}$, $\overline{LDS}$, and $\overline{UDS}$ control signals from the two processors. The $\overline{CSx}$ Latched signal from the arbitration circuit determines which set of control signals are presented to the MC68B10 RAMs.

The above described scheme could allow expansion to 256 words of dual-ported RAM simply by using the available A8 line on the 74LS244s (U12 and U18) shown in Figure 3, plus larger memories. Expansion to larger DPR memory space would require additional address buffers and larger memories.

TIMING CONSIDERATIONS

The timing signals related to the arbitration circuit are shown in Figure 4. This diagram illustrates the special case of chip select signals from two different processors requesting simultaneous access. As will be described, when the simultaneous requests are made, only the $\overline{CS}$ signal of one processor will be allowed to propagate through to form the $\overline{CSx}$ ($\overline{CS1}$, $\overline{CS2}$) Latched signal. In the diagram of Figure 4, the chip select requests are both already asserted before the falling edge of the 12.5 MHz arbitration clock. Since the $\overline{CS2}$ signal is clocked on the falling edge in Figure 4, it will be allowed to propagate through. Note that the Q output for $\overline{CS1}$ (point C) does fall but the signal is not allowed to be clocked into the debounce latch (B1) because it is now held in preset by the $\overline{CS2}$ Latched signal. The complement of the $\overline{CS2}$ Latched signal is also presented to the OR gate at the input of the first latch for $\overline{CS1}$ causing the D input to go high, and on the next positive

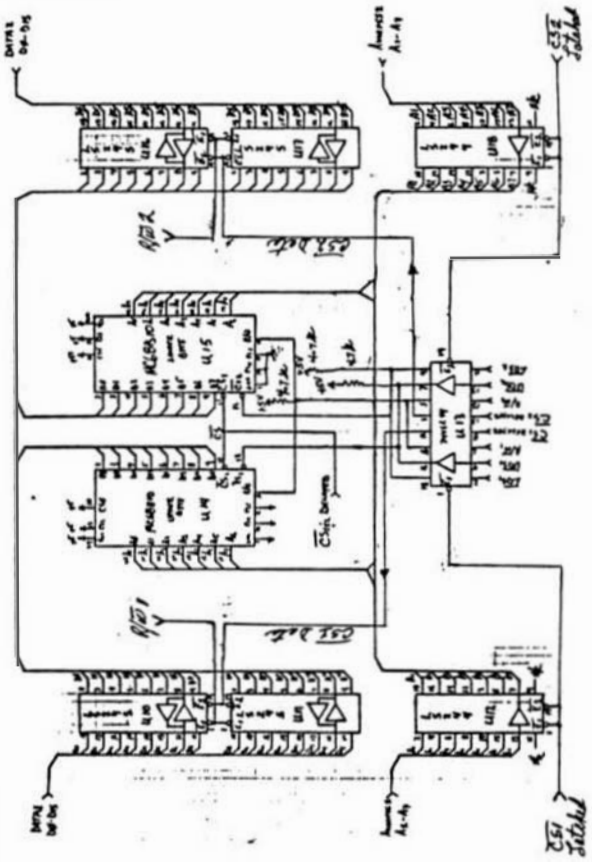


Figure 3. DPR RAM and Buffers Schematic Diagram

edge clock, the waveform at C goes high. When $\overline{CS2}$ is removed, the input latch (waveform D) and the debounce latch for $\overline{CS2}$ are driven high on consecutive negative clock edges. This action removes preset from B1 and allows $\overline{CS1}$ to propagate through on positive clock edges to form the $\overline{CS1}$ Latched signal.

As shown in Figure 3, the $\overline{CS1}$ or $\overline{CS2}$ Latched signal enables the address buffer for one of the MCM68B10s. The $\overline{CS1}$ or $\overline{CS2}$ Latched signal is also presented to the input of one of the 74LS164 shift registers, and a clock period later the $\overline{CS1}$ or $\overline{CS2}$ Delayed signal is presented to the $\overline{CS}$ input of both MCM68B10s. This delay ($\overline{CS1}$ or $\overline{CS2}$ Latched to $\overline{CS1}$ or $\overline{CS2}$ Delayed) is necessary to accommodate the address setup time for the memories.

The read and write cycle timing is shown in Figure 5 (starting with the $\overline{CSx}$ Latched signal). For a read cycle, data is valid 250 nanoseconds after addresses are valid. The $\overline{DTACK}$ signal for the processor (8-MHz MC68000) is asserted a clock period after the $\overline{CSx}$ Delayed signal goes low. Data is then valid approximately 50 nanoseconds later (this sstisfies the $\overline{DTACK}$ low to data setup time for the MC68000L8 specification). For a write cycle, data is valid before $\overline{CSx}$ delayed; therefore, the only specifications that must be met are t_{cyc} for addresses and t_{cs} for the $\overline{CSx}$ Delayed signal. Asserting $\overline{DTACK}$ at the same time as a read cycle sssures that both these specifications are met. The last spectification to be met is the data hold time for the memories. This 10 nanosecond hold time is met by passing the $\overline{CSx}$ Delayed signal through control multiplexer U13 to form the $\overline{CSx}$ Data signal. The $\overline{CSx}$ Data signal will negate approximately the same time as the $\overline{CS}$ signal at the memories; however the buffer delays (U10, U11, U16 and U17) will provide the required data hold time.

TEST SOFTWARE

Once both processors are able to read and write to all DPR (dual-ported RAM) locations, it is necessary to check the arbitration circuitry during real time processing. Two test programs are included. The first program (listing 1 in Figure 6) causes both processors to execute a loop in which they request access to the DPR. Once access is allowed, the processor sets all DPR memory locations to a known value, and then checks to see if all locations retained this value. Completion of the loop results in a message (error or successful completion) and the processor executes a short delay loop to allow the other processor to complete a TAS instruction. The other processor then executes the same routine with the exception of writing a different value to the DPR locations.

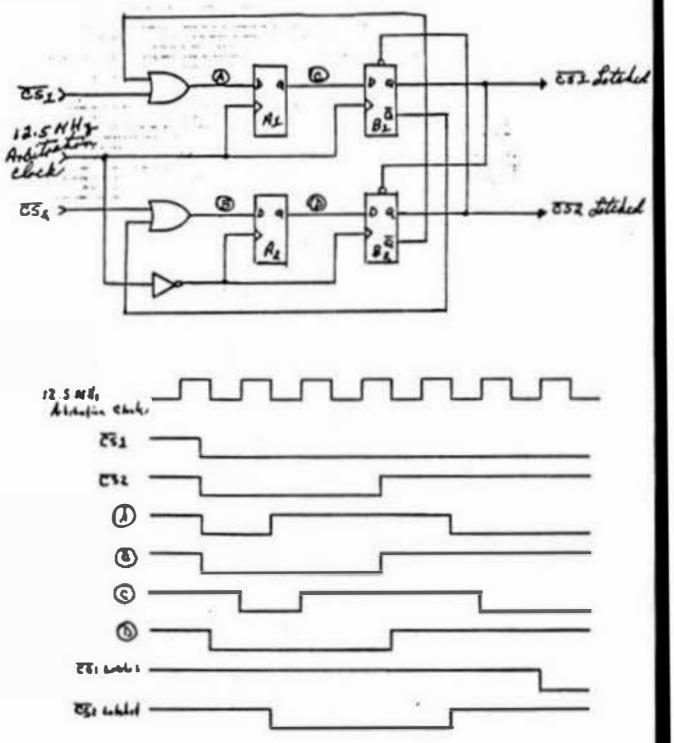
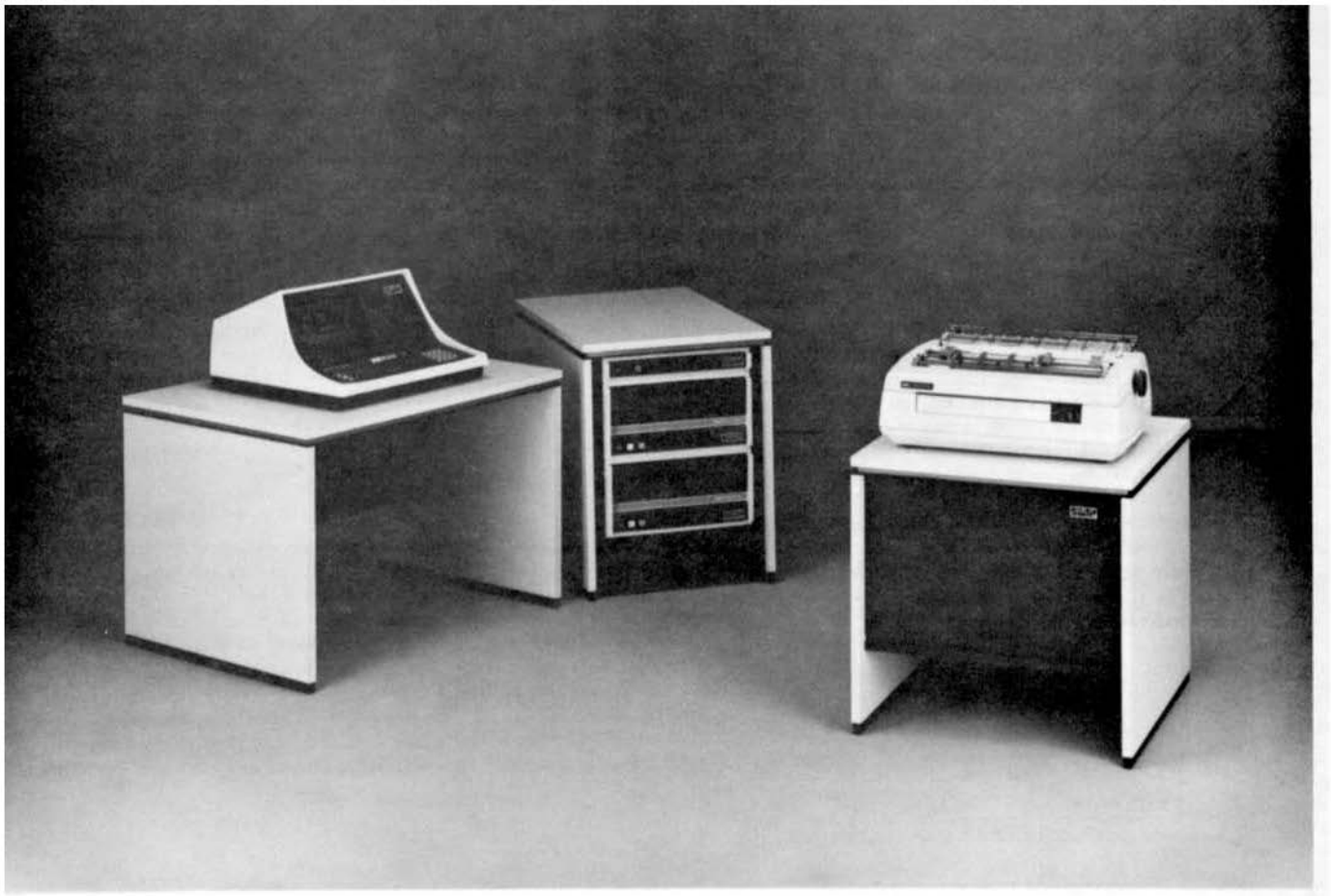


Figure 4. CS Arbitration Timing Diagram



THE COMPLETE BUSINESS SYSTEM

+Multiuser+Highly Expandable+Cost Effective

S+ THE CONCEPT

The S+ system is a modular computer system in which all portions of the hardware and software are designed to work together in the most efficient way possible. An S+ single user system with floppy disk storage is a competitive and cost effective entry level system. Unlike most other small computers being sold as "personal", or "small business" machines, the S+ system may be expanded to maximum capabilities using this same hardware and software. You cannot end up with a DEAD END system that cannot be expanded and whose software is not compatible with larger machines. A basic S+ system may be expanded to thirty-two users, a megabyte of main memory and hundreds of megabytes of hard disk storage by simply plugging in, or connecting the desired upgrade equipment.

TOTAL DESIGN–Hardware and Software

The S+ system is an integrated hardware and software design. The two complement and enhance each other in this system. The UniFLEX® operating system used in the S+ systems is patterned after the Bell Laboratories UNIX® operating system, one of the most admired and widely used operating systems in the world. Instead of being an afterthought, the software is part of the design of the S+ system. You can be sure that with this approach that all parts of the computer operate with maximum efficiency and cost effectiveness.

THE CENTRAL PROCESSOR

The basic S+ system is configured with 256K bytes of memory and can be expanded to more than 1 million bytes. An efficient and fast hardware memory management system is used to allocate the available memory among the users on a dynamic basis. As little as 8K bytes, or the entire memory–if needed–can be used by any individual user. This makes it possible to run very large programs on the system, but it also uses no more memory than necessary for a particular job. The increase in cost effectiveness of this system over crude and outdated bank switching arrangements is dramatic.

The central processor runs in both user and supervisor states. It can detect and reject a defective user program. It is impossible for a user program to go bad and stop the entire system, as can happen quite easily in less sophisticated systems.

Task switching is accomplished by use of a multiple map RAM memory, with sixty-four individual task maps. Each task can access from 4 to 64 K-bytes of memory. Multiple tasks may be used in programs that require more than 64K bytes of memory for execution. When a task is completed the memory is automatically released for other use.

SOFTWARE

The S+ operating system, UniFLEX® is a multiuser, multitasking operating system based on the UNIX® operating system that has been used for many years on Digital Equipment Corp. PDP-11 series minicomputers. It is considered one of the most sophisticated and "user friendly" operating systems available. Variations of UNIX® are rapidly becoming standard on mini and larger microcomputers.

A large variety of languages are available for use with the system. These include FORTRAN, COBOL, BASIC, and Pascal. Word processing packages are also available to give you full text processing capability on the system.

Applications programs are available in large quantities in many fields. This includes general business, medical, dental, veterinary, library and real estate management; plus others. Since the system is multiuser it can also be connected to cash registers to produce a point-ofsale terminal system combined with the computer. The possibilities for application of this system are endless.

THE I/O SYSTEM

The S+ system is totally interrupt driven. All terminal and printer I/O devices connect to an I/O bus separate from the main bus. Up to thirty-two separate devices may be connected to the I/O bus at any one time. If I/O activity is great enough to cause an unacceptable slowdown in system operation, a separate I/O processor can be installed in the system. This plug-in option removes all I/O handling overhead from the main processor and allows operation of up to thirty-two external devices at 9,600 baud. Without an integrated total design, as in the S+ system, it would become impractical to use a UNIX®type operating system in a situation with heavy terminal I/O activity.

DISK STORAGE

A wide range of disk storage capacity is available for the S+ system, from 2.5 M-byte floppy disks to an 80 M-byte Winchester and many sizes between. All disk controllers use direct memory access (DMA) type operations to maximize data transfer and to minimize overhead on the main processor. The Winchester disks also use intelligent controllers along with DMA transfers to preserve the performance that these type devices are capable of giving. Without this distributed intelligence the system performance would be greatly degraded. The UniFLEX®operating system is designed to work at maximum efficiency with this type disk system. The data transfer rates achieved by this combination rival those of large minicomputers.

COMMUNICATIONS

A high speed local network communications system is available to interconnect S+ systems. The VIABUS® network will allow communication between systems at data rates of over 400K baud. Such a system makes it possible to share data between local systems in an efficient and low-cost manner.

AVAILABLE SOON

Tape backup—20M-Byte in less than 15 minutes on a standard ¼ inch cartridge.

Mini-Wini—5 and 10 M-Byte Winchesters—5¼ inch package. Winchester performance, for smaller systems in a small package. UniFLEX® compatible design.

Large Capacity—190 and 340 M-Byte Winchesters, plus SMD cartridge drives.

UniFLEX is a registered trademark of Technical Systems Consultants, Inc.

UNIX is a registered trademark of Bell Labs.

VIABUS is a registered trademark of Southwest Technical Products Corporation.

Two other programs (listing 2 and listing 3) provide examples of messages
sing from a master to a slave processor and vice-versa. These programs are
wn in Figure 7 (for the master routine) and Figure 8 (for the slave
tine). The master processor takes characters sent from a terminal and
res them in a buffer until a carriage return is input. The master MPU then
s the semaphore flag , transfers the message from the buffer to the .
n exiting the DPR, the master MPU sets a message flag (in the low byte of
semphore register) to indicate a message is ready for the slave. The
ve continually checks this message flag to determine if a message is
sent, and if so prints the message out on its terminal and resets the
sage flag.

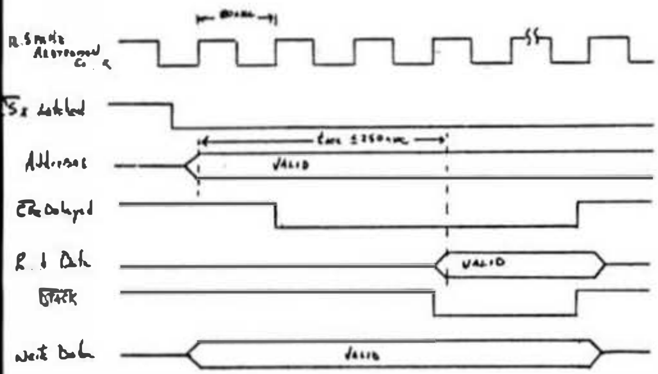


Figure 5. Read/Write Cycle timing Diagram

```
*
*         DELAY SUBROUTINE
*
          MOVE.W   #45,D5          COUNT
DLOOP     NOP
          DBRA     D5,DLOOP        LOOP UNTIL FINISHED
          RTS
          END

**********************************************************************
*  THIS ROUTINE TESTS THE DUAL PORT RAM ARBITRATION CIRCUIT.
*  [illegible]
*  SHOULD BE INITIALIZED AS WELL AS THE VALUE USED FOR THE CHECK FLAG.
**********************************************************************
          ORG      [illegible]
DPRBASE   DS.L     1      ADDRESS OF THE DUAL PORT RAM. THIS VALUE SHOULD
                          BE INITIALIZED BEFORE THE PROGRAM IS
                          STARTED.
[illegible]
*         INITIALIZATION
START     [illegible]     GET ADDRESS OF DPR IN A0
                          SET MEMORY POINTER
                          INC FOR SEMAPHORE
                          GET FLAG
*         MAIN PROGRAM
ALLOC     [illegible]     DPR AVAILABLE?
                          COUNTER
                          SET POINTER
WLOOP     [illegible]     WRITE A WORD
                          ALL DONE?
                          RESTORE COUNT
                          RESET INDEX
CLOOP     [illegible]     CHECK MEMORY
                          ALL DONE?
                          REPORT ERROR
                          RESET SEMAPHORE
[illegible]
                          SET COUNT
                          LOOP UNTIL TIME OUT
```

```
*  THIS PROGRAM SENDS A MESSAGE FROM THE MASTER TO THE SLAVE
*  PROCESSOR. CHARACTERS ARE ENTERED ON THE MASTER CRT UNTIL A
*  CARRIAGE RETURN IS DETECTED. AFTER THE CR THE MASTER ROUTINE
*  TRANSFERS THE MESSAGE FROM A BUFFER TO THE DPR AND SETS A FLAG.
*  THE SLAVE CONTINUALLY READS THE FLAG UNTIL IT IS SET THEN THE
*  [illegible]
*
*         THIS IS THE MASTER ROUTINE.
*
          ORG      [illegible]
COUNT     DS.W     1      NUMBER OF CHARACTERS TRANSFERED
CHARBUFF  DS.B     [illegible]    TEMPORARY STORAGE FOR TRANSFERED CHARACTERS
[illegible]
START     [illegible]     INITIALIZE COUNTER
                          SET POINTER
                          SET UP #7 FOR TRAP CALL TO INPUT ROUTINE
                          CALL INPUT ROUTINE ("TUTOR" MONITOR - TR)
                          SET FOR ECHO CHAR BACK TO TERMINAL
                          CALL OUTPUT ROUTINE ("TUTOR" MONITOR - TR)
                          END OF STRING?
                          GET STRING
                          SAVE CHARACTER
                          INC COUNT
                          GET NEXT CHARACTER
                          LINE FEED
                          OUTPUT LF
                          CALL OUTPUT "TUTOR"
                          DPR AVAILABLE?
                          NOT YET
[illegible]
*         DELAY SUBROUTINE
          MOVE.W   #45,D5          SET COUNT
[illegible]

*  THIS PROGRAM SENDS A MESSAGE FROM THE MASTER TO THE SLAVE
*  PROCESSOR.
*
*         THIS IS SLAVE ROUTINE.
*
          ORG      [illegible]
[illegible]
```

BIT Bucket

Get 6K More RAM from Motorola Micromodule 19 OS-9 System

There could be [illegible] worth of RAM [illegible] a MM19 OS-9 syst[illegible] which is never used if 56K ($0-$E7FF) of RAM is installed. The MM19 has 8K of RAM on-board at $E000 thru $F7FF, but OS-9 stops sizing RAM [illegible] at $D000 due to prom sockets U29 [illegible] U30 not being disabl[illegible] (the Microware installation guide doesn't mention th[illegible]). If the soc[illegible]ets are disabled, OS-9 stops sizing at $E000 due to an internal limit in

the P1 Kernal ROM. A quick MDIR E command will show which memory is being used.

To get 6K more RAM, do the following steps:

1. Cut the following E traces on header K1B to disable PROM 3 (U29) and PROM 4 (U30):

   K1B: pins 25 and 26
   pins 27 and 28

2. Change the following bytes in the Microware P1 Kernal ROM:

offset	new	was
6A4	E8	E0
7AF	F2	2E
7B8	2F	42
7D1	F2	48

3. Change the /t1 and /p1 device device descriptor modules so that there will be no address conflict with your newly gained RAM. Change the actual hardware address in the bytes too. Consult the Debug, Save, Verify, Format and OB9gen commands before proceeding as follows:

device	new address	old address
/t1	$ECB4	$E300
/P1	$ECA0	$E400

Use Debug to link to the device and change the device address at offset $F and $10 to the suggested new address shown above. Exit Debug and use the Save command to save the modified device descriptor. Use the Verify command with the U option to make a new device descriptor file with a correct CRC value. Then use Format and OB9gen to make a new O/S disk using the new device descriptor modules with the correct CRC value.

It should be noted that the Microware's OB-9 for the MM19 can also be used on a Microsolutions I2. The MM17 is cheaper and has a second ACIA port at $ECB4, but its parallel port is for industrial I/O. A custom made printer cable will allow it to be used for the /p port.

Motorola Microsystems
Peter B. Gilmour 04168
2900 Diablo Way
Tempe, AZ 85281
(602) 829-3458

"CLEAR ANY 6800 MEMORY MAP

(Totally)"

by Don Aldridge, Motorola Inc., Semiconductor Products Sector

The following program is the ultimate in memory clear programs. It clears the entire memory map of the 6800 Microprocessor. The catch is, the memory map for the 6800 must be completely filled with 65536 bytes of RAM. The program and the restart vectors must be loaded into the appropriate RAM locations by an external means. The program is loaded at address $0000 and has a start execution address of $0004.

```
0000   36       PSHA
0001   7E 0000  JMP $0000
0004   8E FFFF  LDS #$FFFF
0007   86 00    LDA A #$00
0009   2  F5    BRA $0000
```

The first two instructions to be executed initialize the stack pointer to $FFFF and the A accumulator to $00. The BRA instruction branches back to the beginning of a loop. This loop clears memory locations starting at memory location $FFFF and down through memory by pushing the contents of A accumulator to the memory location pointed to by the stack pointer. The stack pointer is automatically decremented by this instruction.

This operation continues at the stack pointer is decremented. When the stack pointer is decremented to the last byte of the program it is cleared with no effect on program operation since the byte is not in the PSHA, JMP loop. The remaining bytes of the LDS, LDS and BRA instruction are cleared. However, once again there is no effect on program operation.

Next, the last byte of the JMP instruction is cleared. It was already zero, thus no change in program operation. Likewise clearing the second byte of the JMP instruction does not affect program operation. Now the $7E for the JMP instruction is cleared. The memory map now contains only the $36 for PSHA and zeros in the remaining memory locations. At this point the PC is pointing to memory location $0001 where a $00 is located. A $00 appears as a no operation execution to the 6800 and only increments the PC by one. (NOTE: The defined NOP instruction is $01.) The series of $00's are executed until memory location $FFFF is encountered. The PC is then incremented to $0000 where the PSHA is again encountered. The stack pointer is pointing to memory location $0000 so that the PSHA clears the last byte of the program.

The memory map NOW contains all zeros! The program may not be useful but is lots of fun.

Meteorology Dept.
University of Utah
Salt Lake City, Utah
December 14, 1982

Dear Don:

I have enc'osed two machine language subroutines for the TRS-80 COLOR COMPUTER that may be useful to readers.

The first subroutine will restore a program which has been lost to a 'NEW' command, providing the graphics area of memory hasn't been changed in size.

The second subroutine is handy when you want to determine the destination of the outputted data at run time. This routine gives the choice between the RS-232 interface and the Video monitor. It is very fast; a 10k program takes less than 1/4 second to modify.

Sincerely,
David G. McDonald

David G. McDonald

```
                        0120 *                          *
                        0130 * This machine language    *
                        0140 * routine will recover a   *
                        0150 * BASIC program which has  *
                        0160 * been lost by using a     *
                        0170 * 'NEW' command.           *
                        0180 *                          *
                        0190 *                          *
                        0200 ****************************
                        0210 *
                        0220 *
              0019      0230 LBASIC  EQU   $19
              0018      0240 SIMPLE  EQU   $18
              0010      0250 ARRAY   EQU   $10
                        0260 *
                        0270 *
7FE0                    0280 *       ORG   $7FE0
                        0290 *
                        0300 *
                        0310 *  Search for beginning of second line
                        0320 *
7FE0 9E   19            0330 START   LDX   LBASIC
7FE2 C6   04            0340         LDB   #$04
7FE4 3A                 0350         ABX
7FE5 A6   80            0360 NEXTST  LDA   ,X+
7FE7 26   FC            0370         BNE   NEXTST
7FE9 AF   9F 0019       0380         STX   [LBASIC]
                        0390 *
                        0400 *  Search for end of program
                        0410 *
7FED EC   84            0420 PRGEND  LDD   ,X
7FEF 27   04            0430         BEQ   ENDPRG
7FF1 AE   84            0440         LDX   ,X
7FF3 20   F8            0450         BRA   PRGEND
                        0460 *
                        0470 *  Reset Variable Pointers
                        0480 *
7FF5 C6   02             490 ENDPRG  LDB   #$02
7FF7 3A                 0500         ABX
7FF8 9F   18            0510         STX   SIMPLE
7FFA 9F   10            0520         STX   ARRAY
7FFC 39                 0530         RTS
              0000      0540         END

0000 TOTAL ERRORS

ARRAY   0010
ENDPRG  7FF5
LBASIC  0019
NEXTST  7FE5
PRGEND  7FED
SIMPLE  0018
START   7FE0
```

```
                        0120 *  Output Devices in a   *
                        0130 *    executing BASIC     *
                        0140 *       program          *
                        0150 *                        *
                        0160 *     Called using:      *
                        0170 *      A$=USR0(A$)       *
                        0180 *                        *
                        0190 *  A$= "0"  for Monitor  *
                        0200 *      "2"  for Printer  *
                        0210 *                        *
                        0220 ******************************
                        0230 *
                        0240 *
              0019      0250 LBASIC  EQU   $19        * Bottom of BASIC Program
              0018      0260 TBASIC  EQU   $18        * Top of BASIC Program
              8723      0270 PRINT   EQU   $8723      * Token + ASCII for "PRINT@"
              AC31      0280 CASSET  EQU   $AC31      * ASCII for "-1" (CASS OUT)
                        0290 *
                        0300 *
7FE0                    0310         ORG   $7FE0
                        0320 *
                        0330 *
7FE0 10AE 02            0340 START   LDY   2,X        * Load Addr of Device No. (A$)
7FE3 9E   19            0350         LDX   LBASIC
                        0360 *
7FE5 CC   8723          0370 SEARCH  LDD   #PRINT     * Check for "PRINT@"
7FE8 10A3 80            0380         CMPD  ,X+
7FEB 27   05            0390         BEQ   CASSCK
                        0400 *
7FED 9C   18            0410 TEST    CMPX  TBASIC     * Check for end of program
7FEF 26   F4            0420         BNE   SEARCH
7FF1 39                 0430         RTS
                        0440 *
7FF2 CC   AC31          0450 CASSCK  LDD   #CASSET    * Check for "-1" (CASS OUT)
7FF5 10A3 01            0460         CMPD  1,X
7FF8 27   F3            0470         BEQ   TEST       * Bypass if output to cassette
                        0480 *
7FFA A6   A4            0490         LDA   ,Y         * Load Device Number (ASCII)
7FFC A7   02            0500         STA   2,X        * Change Output Device
7FFE 20   E0            0510         BRA   TEST
                        0520 *
              0000      0530         END

0000 TOTAL ERRORS

CASSCK  7FF2
CASSET  AC31
LBASIC  0019
PRINT   8723
SEARCH  7FE5
START   7FE0
TBASIC  0018
TEST    7FE0
```

John Farnald
255 Eagleview Pl.
Newbury park, CA 91320

Editor
68 Micro Journal
5900 Cassandra Smith Rd.
Hixson, Tennessee 37343

Dear Sir:

I am a TRS-80 Color Computer owner and have been using it, together with the Radio Shack Disk system and Line Printer VII for about 10 months. For about 8 of those 10 months my computer has been used on a near daily basis for a number of projects related to my work as a Pilot for a large government agency.

During early November I was deeply involved in the computerization of all the maintenance requirements (scheduled and unscheduled) on each of our eight assigned aircraft. This project, coupled with several other equally critical ones placed heavy demands on my normally reliable equipment. Unfortunately, like any piece of electrical hardware given enough utilization, a failure point was reached. The initial indication was that the disk controller had failed and all else was OK. This did not prove to be the case and in fact a failure in the computer had apparently induced a failure in the disk controller.

This unforeseen malfunction immediately put my maintenance support activities on hold and due to a heavy demand for aircraft support, ill effects rapidly developed.

A visit to my local (Ventura, CA) Radio Shack Computer Center produced results that truly astonished this writer. On my initial visit the Disk Controller was replaced by a "loaner" to help me stay on schedule while my unit was being repaired. An attempted use of this controller immediately rendered it inoperative, as well as my 32K computer. An immediate trip to the "Shack" produced a computer circuit board change and the receipt of my newly repaired Disk Controller, all in less than one hour. As if the quick and thoughtful service wasn't enough, they didn't even whimper when I returned their "blown" Disk Controller.

Since my computer was no longer under warrantee a one time (and a reasonable one) charge for this service was made. The Disk Controller and my 32K unit are now under a 45 day warrantee and I am back on schedule in my projects.

Radio Shack should be proud of their stores that "go the extra mile" for customers, as these establishments are the computer owner and buyer's most direct link with the manufacturer. If I ever had thoughts of changing to another brand of computer, service like that which I have just described would certainly go a long way towards convincing me to stay with the "BEST".

Sincerely,

PRODUCT ANNOUNCEMENT

MCOMMAND

MCOMMAND(tm) is a Flex(tm) utility that will convert any disk resident command to a memory resident command. As an example of its use, executing MCOMMAND,CAT will create a memory resident command called CAT that does the same thing as the disk utility CAT. As another example, executing MCOMMAND,LIST,COPY,ASN will create three memory resident commands that perform identically to the standard Flex utilities LIST, COPY and ASN. There is no restriction on the type of disk resident command that can be converted to a memory resident command.

MCOMMAND provides a substantial enhancement to single drive systems, where it makes it unnecessary to keep copies of common utilities on every working disk. Even in multiple drive systems, MCOMMAND can be used to eliminate a certain amount of disk juggling, as in copying from one disk to another, where neither disk contains the COPY command. On any system, memory resident commands execute almost instantaneously since they do not have to be retrieved from disk.

Two additional utilities, CATALOG and DROP, are supplied with MCOMMAND. CATALOG displays a list of memory resident commands. It also displays the amount of memory occupied by each command. DROP allows you to remove memory resident commands in order to make more space available for user programs.

In any operating system, there is an inevitable tradeoff between size on the one hand, and convenience on the other. MCOMMAND and DROP essentially allow you to make this tradeoff for yourself.

A pricing policy for MCOMMAND has not been established. Dealer inquiries are invited. Further information and a list of dealers may be obtained from James Arbuckle, P.O. Box 328, Ambler, Pa. 19002 (215:643-0788).

November 26, 1982

Computer Publishing Center
68' Micro Journal
5900 Cassandra Smith
P.O. Box 849
Hixson, Tenn 37343

Sir:

I've enclosed the following two computer programs for possible inclusion in your magazine. Both programs do exactly the same function, except one is written in BasicO9 and the other in Pascal. Both programs are for Microware's OS-9 DOS. The Pascal program is specifically written to use Microware's Pascal programming package.

After I had written the BasicO9 version, I found that I was using the program often, thus I rewrote the program into Pascal so that I could compile the program into native code, and use it as a utility program. If any variations are needed I simply modify the BasicO9 version and use it for printing text. Variations would include printing a top of page header message, or copyright info.

I like the OS-9 DOS a lot. The BasicO9 package is outstanding. I find that after one masters BasicO9, there is not much sense in writing programs in other languages. It is a wonderful language to work with. Its speed is excellent, and except for those applications that require high speed, BasicO9 is the language to use.

Microware's Pascal program package is probably the best one on the market for anyone. I own several Pascal compilers and have used them all. I find that Microware's Pascal is the easiest to use and develop programs with. What I like most about Microware's Pascal is the Virtual Memory Pcode Interpreter. This program package makes writing giant programs extremely easy, as one is not limited to the available user RAM, one is limited only on the disk space that is available, and with a hard disk the speed is excellent.

As for hardware, several months ago I bought one of the GIMIX 19mb hard disk systems. This microcomputer is very good. The hard disk is very fast and the overall system performance has been outstanding. Gimix, themselves deserve a lot of credit in getting their product working, its worth the price. Gimix's field service department gets my blessing too, as once I had a problem with the hard disk itself, it took three days from the day I expressed shipped the unit back to them, until I recieved a replacement from GIMIX. The one week typical service turnaround that GIMIX provided can't be beat, for keeping downtime to an absolute minimum. Other than the hard disk acting funny once, I've never experienced any troubles at all with the system, it has worked flawlessly. I recommend the Gimix systems for anyone who wants a high powered microcomputer system, they're very good and reliable machines.

When the Gimix system was set up I used the ADDS Viewpoint 3A CRT terminals, these terminals are an excellent product, they provide for all the basic needs and requirements that one could want. Both terminals I've got have worked without any problems at all. They're very nice terminals. I've also have connected a ANADEX 9620 printer to the system, using the serial port feature. This printer uses a MC6809 processor too (it would be nice if everything used 6809's), and can print 5 different character fonts and has dot graphics also. It has proven to be an excellent printer for use with the system, and is very quiet in operation too.

My old computer is a PERCOM based 6809 system. It uses the Percom SBC-9 CPU and Percom's LFD-400 controller and MPI 40 trk drives. I modified it to use 56k bytes of ram. It also has code that allow it to run TSC's Flex and Percom's MPX-9 DOS, and my favorite Microware's OS-9 DOS (I've modified OS-9 to work with 56k bytes of ram on the Percom system).

The PERCOM system is a nice system, but I've found that one has to from time to time reseat chips, and reseat boards, etc. Even the disk drives require reseating the cables and connectors. It tends to work for a while, then flakes out, almost invariably reseating something cures the problem. I hope the other manufacturers try and use good IC sockets and connectors, as I hated to have spent several months looking for an imaginary intermittent problem. Other than having to reseat things from time to time, the PERCOM stuff has worked fine. It is somewhat of a shame that they went for hard sector disks instead of soft sector disks. As everyone else uses soft sectoring, making the system pretty much a loner in the world. The Percom system is not bad, it does work, but it is cantankerous.

I recently ordered a new CPU board and disk contoller for upgrading the old computer to newer GIMIX DS/DD disk drives, using OS-9. The Percom cpu cannot handle DMA accesses properly, thus if you want DMA you'll have to get a new CPU board. After using the less floppies on the GIMIX system, you get spoiled, thus the reason for the upgrade.

Earl W. Bollinger
18810 North 91st. Ave., Space 16
Peoria, Arizona 85345

```
Page   1  00/00/00 00:00:00  OS-9 Pascal - release 1.c.1
STMT  PLOC LEV
----+----+----1----+----2----+----3----+----4----+----5----+----6
----+----7----+----8----+----9----+----10---+----11
   1     00  0 ( This program is used to output a text file to the printer. )
   2     00  0 ( It forces form feeds and prints page numbers, starting with)
   3     00  0 ( the page number as entered. Also, if a "^cr" is encountered)
   4     00  0 ( a page number and form feed will be outputted.            )
   5     00  0 ( By E.W.Bollinger on Aug. 21, 1982      ver 1.0            )
   6     00  0 ( 18810 N. 91st. Ave., Peoria, Arizona 85345                )
   7     00  0 PROGRAM PAGEINT;
   8     00  0
   9     00  0 VAR
  10     00  0   F: File of char;
  11  -1430  0   Str: array[1..40] of char;
  12  -1050  0   C: array[1..120] of char;
  13  -3130  0   FF: char;
  14  -3140  0   I,Lines,Page,Pnum: integer;
  15  -3220  0   Printer : text;
  16  -6570  0   Term : text;
  17  -6120  0
  18  -6120  0 BEGIN
  19      0  1   Pcnt:=58; (number of lines to a page )
```

```
 20   52  1   FF:=chr(12); {form feed symbol}
 21   57  1   Lines:=0; {used to count the number of lines on a page}
 22   61  1
 23   61  1   Rewrite(Printer,'/P1 ');
 24   72  1   Rewrite(Term,'/TERM ');
 25   83  1
 26   83  1   Writeln(Term);
 27   88  1   Writeln(Term);
 28   93  1   Writeln(Term,'Page Number Printing Program            Ver. 1.0')
;
 29  110  1   Writeln(Term);
 30  115  1   Write(Term,'Enter filename (include pathlist): '); Prompt(Term
);
 31  132  1
 32  132  1   I:=1;
 33  136  1   WHILE (I<40) AND (not eoln) DO
 34  150  2     BEGIN
 35  150  2       Read(Str[I]); I:=succ(I);
 36  173  2     END;
 37  176  1   Readln;
 38  179  1
 39  179  1   WHILE I<=40 DO
 40  188  2     BEGIN
 41  190  2       Str[I]:=' '; I:=succ(I);
 42  211  2     END;
 43  214  1
 44  210  1   Reset(F,Str);
 45  225  1
 46  225  1   Writeln(Term);
 47  230  1   Write(Term,'Enter the starting Page Number: '); Prompt(Term);
 48  247  1   Read(Pnum);
 49  253  1   Readln;
 50  258  1   Writeln(Term);
 51  263  1
 52  263  1   WHILE not eof(F) DO
 53  272  2     BEGIN
 54  272  2       FOR I:= 1 TO 128 DO
 55  292  3         BEGIN
 56  292  3           C[I]:=' '
 57  305  3         END;

Page   2  00/00/00 00:00:00  OS-9 Pascal - release 1.c.1
STMT  PLOC LEV
-----+-------+-+----------1---------2---------3---------4---------5---------6----
-----7---------8---------9---------10---------11
 58  328  2       I:=1;
 59  332  2       WHILE (I<128) and (not eoln(F)) DO
 60  349  3         BEGIN
 61  349  3           C[I]:=F^; Get(F);
 62  371  3           I:=succ(I);
 63  378  3         END;
 64  381  2       Readln(F);
 65  386  2       Lines:=succ(Lines);
 66  393  2       IF (C[1]='^') and (C[2]=' ') THEN
 67  421  3         BEGIN
 68  421  3           FOR I:= Lines TO Page DO
 69  443  4             BEGIN
 70  443  4               Writeln(Printer)
 71  443  4             END;
 72  460  3           Writeln(Printer);
 73  473  3           Writeln(Printer, 'Page ':36, Pnum);
 74  498  3           Write(Printer,FF); Prompt(Printer);
 75  519  3           Lines:=0;
 76  523  3           Pnum:=succ(Pnum)
 77  527  3         END
 78  530  3       ELSE
 79  533  3         BEGIN
 80  533  3           IF Lines=Page THEN
 81  543  4             BEGIN
 82  543  4               Writeln(Printer,C);
 83  562  4               Writeln(Printer);
 84  567  4               Writeln(Printer, 'Page ':36, Pnum);
 85  592  4               Write(Printer,FF); Prompt(Printer);
 86  606  4               Lines:=0;
 87  610  4               Pnum:=succ(Pnum)
 88  614  4             END
 89  617  4           ELSE
 90  620  4             BEGIN
 91  620  4               Writeln(Printer,C)
 92  634  4             END;
 93  639  3         END;
 94  639  2     END; {of the while loop}
 95  642  1
 96  642  1   FOR I:= Lines TO Page DO
 97  664  2     BEGIN
 98  664  2       Writeln(Printer);
 99  669  2     END;
100  689  1   Writeln(Printer);
101  694  1   Writeln(Printer,'Page ':36, Pnum);
102  719  1   Write(Printer,FF); Prompt(Printer);
103  733  1   Writeln('The computer is done printing now');
104  746  1   Writeln;
105  749  1 END.

PROC NAME      PSEC  PSIZE  LOCAL  STACK   CREC  CSIZE  DEBUG
   0 PNPRINT      1      7    614     21      4    213      0
                       766    614     21           213

105 Lines of source code compiled with no errors found
```

```
LOCAL=3675
EXTENDED=000
HEAP=16384
ACTUAL HEAP=2698
ACTUAL STACK=3290
FREE STACK=1292
FREE HEAP=2534
```

```
PROCEDURE PNPRINT
 0000      (* THIS PROCEDURE IS USED TO PRINT HARDCOPY WITH *)
 0033      (* THE TOP OF FORM (FORMFEE ) THROWN IN AT       *)
 0066      (* APPROPRIATE POINTS IN THE LISTING OF TEXT.    *)
 0099      (* IT ALSO PROVIDES FOR THE TEXT TO TRIGGER      *)
 00CC      (* A FORM FEED BY USING A " ^ (CR) " ON A SINGLE *)
 00FF      (* LINE. PAGE NUMBERS ARE PRINTED AT THE         *)
 0132      (* BOTTOM OF EACH PAGE PRIOR TO A FF.            *)
 0165      (* AND THE USER CAN SPECIFY THE PAGE NUMBER      *)
 0198      (* TO START WITH .                               *)
 01CB      (* WRITTEN BY E.W.BOLLINGER, JUNE 15, 1982       *)
 01FE      (* 10810 N 91ST AVE., PEORIA, AZ. 85345          *)
 0231      DIM FILENAME:STRING
 0238      DIM PPATH,FPATH,CHAR:BYTE
 0247      DIM DMY,PAGE,PNUM:INTEGER
 0256      PRINT
 0258      PRINT
 025A      INPUT "FILENAME (INCLUDE PATHLIST)? ",FILENAME
 027F      PRINT
 0281      INPUT "ENTER STARTING PAGE NUMBER: ",PNUM
 02A5      PRINT
 02A7      OPEN #PPATH,"/P1":WRITE
 0285      OPEN #FPATH,FILENAME:READ
 02C1      PAGE:=0
 02C8      SEEK #FPATH,0
 02D1      WHILE EOF(#FPATH)=FALSE DO
 02DD        GET #FPATH,CHAR
 02E7        IF CHAR=94 THEN
 02F3          GET #FPATH,CHAR
 02FD          IF CHAR=13 THEN
 0309            FOR N=PAGE TO 50
 031C              PRINT #PPATH
 0322            NEXT N
 032D            PRINT #PPATH; TAB(35); "PAGE "; PNUM
 0343            PRINT #PPATH,CHR$(12);
 034E            PAGE:=0
 0355            PNUM:=PNUM+1
 0360            GOTO 50
 0364          ELSE
 0368            PRINT #PPATH; "^";
 0373            PRINT #PPATH; CHAR;
 037E          ENDIF
 0380        ENDIF
 0382        IF CHAR=13 THEN
 038E          PAGE:=PAGE+1
 0399          IF PAGE=50 THEN
 03A5            PRINT #PPATH,CHR$(CHAR)
 03B0            PRINT #PPATH
 03B6            PRINT #PPATH; TAB(35); "PAGE "; PNUM
 03CC            PRINT #PPATH,CHR$(12);
 03D7            PAGE:=0
 03DE            PNUM:=PNUM+1
 03E9            GOTO 50
 03ED          ENDIF
 03EF        ENDIF
 03F1        PRINT #PPATH,CHR$(CHAR);
 03FD 50   ENDWHILE
 0404      FOR N=PAGE TO 50
 0417        PRINT #PPATH
 041D      NEXT N
 0428      PRINT #PPATH; TA (35); "PAGE "; PNUM
 043E      PRINT #PPATH,CHR$(12)
 0448      CLOSE #PPATH
 044E      CLOSE #FPATH
 045A      PRINT "I'M DONE PRINTING NOW."
   6E      PRINT
  470      END
```

```
{ This program is used to output a text file to the printer. }
{ It forces form feeds and prints page numbers, starting with}
{ the page number as entered. Also, if a "^cr" is encountered}
{ a page number and form feed will be outputted.             }
{ By E.W.Bollinger on Aug. 21, 1982      ver 1.0             }
{ 10810 N. 91st. Ave., Peoria, Arizona 85345                 }
PROGRAM PNPRINT;

VAR
  F: File of char;
  Str: array[1..40] of char;
  C: array[1..128] of char;
  FF: char;
  I,Lines,Page,Pnum: integer;
  Printer : text;
  Term : text;

BEGIN
  Page:=50; {number of lines to a page }
  FF:=chr(12); {form feed symbol}
  Lines:=0; {used to count the number of lines on a page}

  Rewrite(Printer,'/P1 ');
  Rewrite(Term,'/TERM ');

  Writeln(Term);
  Writeln(Term);
  Writeln(Term,'Page Number Printing Program            Ver. 1.0');
  Writeln(Term);
  Write(Term,'Enter filename (include pathlist): '); Prompt(Term);

  I:=1;
  WHILE (I<40) AND (not eoln) DO
    BEGIN
      Read(Str[I]); I:=succ(I);
    END;
  Readln;

  WHILE I<=40 DO
    BEGIN
      Str[I]:=' '; I:=succ(I);
    END;

  Reset(F,Str);

  Writeln(Term);
  Write(Term,'Enter the starting Page Number: '); Prompt(Term);
  Read(Pnum);
  Readln;
  Writeln(Term);

WHILE not eof(F) DO
  BEGIN
    FOR I:= 1 TO 128 DO
      BEGIN
        C[I]:=' '
      END;
    I:=1;
    WHILE (I<128) and (not eoln(F)) DO
      BEGIN
        C[I]:=F^; Get(F);
        I:=succ(I);
      END;
    Readln(F);
```

```
    Lines:=succ(Lines);
    IF (C[1]='^') and (C[2]=' ') THEN
      BEGIN
        FOR I:= Lines TO Page DO
          BEGIN
            Writeln(Printer)
          END;
        Writeln(Printer);
        Writeln(Printer, 'Page ',136, Pnum);
        Writeln(Printer,FF); Prompt(Printer);
        Lines:=0;
        Pnum:=succ(Pnum)
      END
    ELSE
      BEGIN
        IF Lines=Page THEN
          BEGIN
            Writeln(Printer,C);
            Writeln(Printer);
            Writeln(Printer, 'Page ',136, Pnum);
            Write(Printer,FF); Prompt(Printer);
            Lines:=0;
            Pnum:=succ(Pnum)
          END
        ELSE
          BEGIN
            Writeln(Printer,C)
          END;
      END;
  END; {of the while loop}

FOR I:= Lines TO Page DO
  BEGIN
    Writeln(Printer);
  END;
Writeln(Printer);

  Writeln(Printer,'Page ',136, Pnum);
  Write(Printer,FF); Prompt(Printer);
  Writeln('The computer is done printing now!');
  Writeln;
END.
```

RAM CHECK FOR THE COLOR COMPUTER

BY EMMETT M.LEWIS JR.

ONE of the first plug in rompacks I bought for my Color Computer was the Diagnostics Rompack.This rompack was more than adequate for my 4K ram computer because it would test 16K ram.however after expanding to Extended 32K ram it was not able to test the upper 16K ram.Only two choices remained,wait for Radio Shack to write and market a 32K diagnostics or write it own.my conclusion is obvious,hence this program.

If you have color basic the program will be placed in memory at ram location 1537 and occupy the next 250 bytes.so you may type in the program and run it without additional effort.If you have Extended Basic, the program will be placed in memory beginning at 7680 because of the automatic pclear 4,therefore you must type POKE 25,6:POKE 26,1 and press ENTER before typeing in the program.this will tell the computer where to place the program in memory.We also have to tell the computer where the program ends after the program is entered,type POKE 27,6:POKE 26,234 and press ENTER,otherwise the program will alarm in about byte 7899.

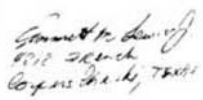
As the value of PK increases the program will begin to slow.if you have 32K ram you may wish to increase the value of PK.Be sure the new value of PK is a few bytes less than the last byte displayed on the screen to insure a complete test.Changing PK is not required.

It should be noted Radio Shacks Diagnostics will test most of the Color Computers functions such as sound,joystick,printer,RS 232,roms, video display generator,keyboard,16K ram,cot.and is recommended for periodic tests.This program is intended as an extension to test each successive ram byte to insure all bits set and reset.provide alarms,plus a visual display of the ram byte number and value contained in the byte being tested.

The start of ram test may be changed simply by supplying a new value of PK.but for the newcomer this alteration if to low could possibly stomp all over the program and result in some confusion for the computer.An easy way to check for this in case of unexpected results is to simply type LIST and press enter,the program should appear on the screen without change.If you have black square dots on the display or changes in the program the value of PK was to low.After the Program is operating it will continue to run until it runs out of ram bytes or crashes due to a defective byte.The last byte tested will appear on the bottom left of the screen and should coincide with the amount of ram your computer has.For example,my computer contains 32K ram and the program alarms in about 32,544 bytes.

One final message,the program will Probably not alarm on exactly the same byte as you near the end of ram.It would be a good idea to keep a record of which byte failed for future reference.

```
5 'Written by Emmett M.Lewis Jr.,4818 Pranon,Corpus Christi,Texas
10 CLS:PK=1780                      'START RAM TEST LOCATION
20 PK=PK+1                          'ADVANCE IN RAM
30 POKE PK,0                        'RESET BITS
40 P1=PEEK(PK)                      'LOOK AT BYTE BEING TESTED
50 IF P1=0 THEN 60 ELSE 120         'CHECK RESULTS
60 POKE PK,255                      'SET BITS
70 P=PEEK(PK)                       'LOOK AT BYTE TEST
80 IF P=255 THEN 90 ELSE 120        'CHECK RESULTS
90 PRINT PK;P;P1                    'PRINT BYTE TESTED,BITS SET,BITS RESET
100 IF P<>255 THEN 120              'DOUBLE CHECK BITS SET
110 GOTO 20                         'END LOOP
120 PRINT PK;P;P1                   'PRINT FAILED BYTE,RESULTS
130 SOUND 50,50:SOUND 230,50        'AUDIABLE ALARM
```



To all SS-50 users:

For some time there have been reports (P. Stark, 68 MICRO, etc) about 'strange' behavior often eventually traced to the MOLEX connectors.

I can only hang my head in shame at not having reported this before, but this bulletin board provides too easy a forum to not act now. Maybe Don Williams can use some of this in 68 MICRO. The problem is that my notes are gone and this discourse is from memory.

As a manufacturer utilizing MOLEX and/or the SS-50 bus in product, we ran into this problem a couple years ago. After successfully beating down the pressure to switch busses or connectors, I had some time for R & D. I talked with MOLEX, METHODE, and other manufacturers of this type of connector. I talked with users like SSB and Richard Don at GIMIX. These conversations do lead to some general conclusions, but first some history.

We were experiencing many reliability problems, mainly with the lab systems, but to some extent with product. This was first diagnosed as cracked traces and/or broken solder joints on the motherboard. Boards soon appeared with 1/2 inch of epoxy on the back side, (talk about stiff) but the problems persisted. I noticed the occasional bent female which made poor contact, but that was not the whole problem. Usually replacing the female half brought temporary relief. You could see dark lines on both parts from the matings, but the female usually looked worse. Since it did not make sense for the female to wear more than the male, I pulled a pin and found a microscope. The dark line we have seen was BARE COPPER! It naturally oxidizes and presto - "morning sickness".

At this point I started talking to other users. Their experience confirmed mine, particularly with the systems that have boards frequently pulled and replaced. But they also reported trouble in systems that had not had a board moved - particularly near salt water. This has since been born out in my experience. We make product that is used in steel mills and iron foundries. The atmosphere there is often acidic and even the tin seems to oxidize and trouble develops.

Per MOLEX, an unlubricated tin contact is good for about 10 to 15 insertions. Lubrication improves this by a factor of 2 or 3. Per METHODE, their square pin gives better life than the round MOLEX, maybe double. While this looks good (projected life of 40 to 90 insetions with METHODE) it does not fix the coastal or steel mill problem. Gold connectors extend the lubed and bare life by a factor of 2, and do not - I repeat, DO NOT - corrode.

We bit the bullet and switched to gold. The problems just do not exist any more. For those who don't like the taste of gold bullets, my tests showed that METHODE's claims are real. When you re-pin your mother board, you could try their square pins. Lubrication with a non-conductive gel may slow the oxidation and certainly prolongs the usefull contact life. We use NYOGEL by the William Nye company in Bedford Mass.

Carl Kreider
22305 CR 28
Goshen, IN 46526
219-875-7019

Editor's Note: The above is one of the many useful items that has been left on the 68 MICRO JOURNAL™ modem system, phone number 1-615-842-6809. And I want to thank Carl for the courtesy of leaving needed information, and at his expense. That is what 68 MICRO JOURNAL is really all about; thousands of different 68XX users, banded together, in a spirit of cooperation and help.

I have on several occasions, in the past, remarked on this very same problem. However, I would caution those who would use the METHODE 'square' pins that they are a 'REAL BEAR' to pull off a motherboard after they are seated. In fact I have seen motherboards broken by the force necessary to remove the square pin types. If they must be removed (and they probably will at some time or another) I would recommend that you pry with a screwdriver blade along the entire row of pins, especially 50 pin boards, section at a time, until the board can be lifted without exerting excessive or destructive pressures.

It would be my recommendation that the use of a good grade contact cleaner be used along with the lubrications indicated in Carl's note. The best solution is gold connectors and 'LEAVE 'EM ALONE!!!

Thanks again Carl and call in often, I and thousands of readers appreciate your helpful input.

OMW - - -

MOTOROLA INC.
MOS Microprocessor Div
3501 ED BLUESTEIN BLVD
AUSTIN, TEXAS 78721
For further information contact

High Speed CMOS + 6809 MPU = A Winning Combination

Austin, Texas, September 20, 1982..... Motorola Microprocessor Division will offer the powerful MC6809 microprocessor in high performance HCMOS. This enhanced model, the MC68HC09E, will be designed in 3 micron technology for high density architecture and reduced power consumption.

In terms of hardware and software, the MC68HC09E is an ideal processor for higher level language execution as well as control applications. External clock inputs allow synchronization with peripheral devices, external systems, or other MPUs. Internal bus buffers are controlled by the tri-state input control signal.

The programming model of the MC68HC09E includes two 16-bit Index Registers, two 16-bit Indexable Stack Pointers, two 8-bit Accumulators that can be configured as a single 16-bit Accumulator, an 8-bit Direct Page Register and an 8-bit Condition Code Register. Multiprocessing applications are made even more efficient by the advanced VMA (AVMA) signal. When true, this signal indicates that the MPU desires bus control in the next cycle. In this type of application, the BUSY status line indicates the need to hold off rearbitration of the next bus cycle to insure system integrity.

The MC68HC09E will be available 1Q84. System development support is available, including OS-9 (trademark of Microware Inc.) and FLEX (trademark of Technical Systems Consultants Inc.)

Announcement Made by Motorola of an 8-Bit HCMOS Microcomputer

Austin, Texas, September, 20, 1982.... Motorola Microprocessor Division announces an advanced microcomputer in HCMOS, the MC68HC04. Its 8-bit architecture is designed using 3 micron technology and contains a CPU, internal clock, 1024 bytes of User ROM, 72 bytes of ROM used for look-up tables, 32 bytes of RAM and 20 bidirectional I/O lines. Other hardware features include power saving STOP and WAIT modes, and self-checking capabilities. A leadless chip carrier is offered as a packaging alternative to a standard plastic or ceramic DIP.

The MC68HC04 has a byte efficient instruction set that is very similar to the M6800 Family and is easily programmed. There are also 10 powerful addressing modes to accommodate the user's needs. There are true bit manipulation capabilities plus bit test and a branch instruction. For user convenience there is also a single instruction for examining or changing the contents of a memory location. MC68HC04 is expected to be available in mid-1983 and will be priced in the $3.00 range in high volume.

Computer Publishing Center
68 MICRO JOURNAL
PO Box 849 59000 Cassandra Smith
Hixson, Tennessee 37343

Dear Mr. Williams:

PRESS RELEASE
Z80 SIMULATOR FOR THE MC6809

I am pleased to announce the availability of a Z80 simulator for the MC6809, and I have enclosed a copy for your most rigorous testing. This will be an important addition to the software toolkits of many programmers. Now, not only will 6809 owners be able to see what all those magazine programs for the Z80 do; but they can even run CP/M® if they want to! At last SWTPC owners can see what they are/are not missing.

I wish to emphasize right here, though, that I did not create a Z80 simulator because I was dissatisfied with the 6809 or with FLEX® in any way. Now that I can make a more direct comparison between the two systems, I have an even greater respect for the instruction set of the MC6809 and the easy operation of FLEX®. With this simulator you can make the comparison too.

You get two simulators for the price of one, as well. The Intel 8080 instruction set is a subset of the Zilog Z80, therefore all the 8080 programs become available too.

At present, the Simulator uses some of the advanced features of the SWTPC CT-82 terminal; calls routines in the SBUG-E monitor; employs a counter in the SWTPC MP-ID interface to simulate a Z80 "refresh" register; and is distributed on a FLEX9® 8" single density disk. If your terminal or monitor is different from these, let me know when you order and I will see if I can modify the Simulator. If you do not employ a MP-ID board, please let me know as well.

The Simulator, in its largest version, exists in memory from $9237 to $C012. It is written in NON-relocatable code, if you feel this will cause you a problem, please let me know. I have found that it is most convenient to have the Simulator co-resident with FLEX®. Unfortunately, I cannot include the source code at this time.

Although notes on the use of the "standard" Single Density CP/M® are included in the documentation, in accordance with the agreement, required by Digital Research, no part of the CP/M®

documentation or their programs are included in any form. The purchaser of the simulator must purchase CP/M" separately from Digital Research. A thorough knowledge of Assembly language techniques, and a maximum dose of dogged determination, will be required.

Purchasers of the Simulator will also know in a few seconds that they have a working program, for a 9k simulator test routine called SIMTEST is included which tests each instruction.

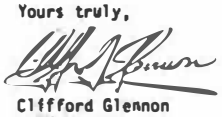
A 1mhz 6809 simulating a 4mhz Z80 can not be expected to break any speed records. Yet I have found that the speed of this simulator is at least adequate.

A copy of the Z80 Simulator can be obtained by sending check or money order in the amount of $60.00 to the undersigned.

CP/M" is the trademark of Digital Research, Box 579
Pacific Grove, California

FLEX" is the trademark of Technical Systems Consultants, Inc.
P.O. Box 2570, West Lafayette, Indiana 47906

Yours truly,



Clifford Glennon
3395 Nostrand Ave. Apt 2
Brooklyn, New York 11229

RON RAINE
7589 OAK ST.
VANCOUVER B.C.
V6P-4A4

```
230 IF F>700 THEN GOSUB 660
240 REM LINES 230 & 280 ARE HELPFUL WHEN MAKING CHANGES
270 REM TO THE PROGRAM. THEY CAN BE ELIMINATED.
280 PRINT TAB(31);"OUTER LOOP=";T;:PRINT"INNER LOOP=";F
290 IF F=700 THEN GOSUB 630
300 IF F=1000 THEN GOSUB  630
310 L=L+1
320 IF L=3 THEN L=0:GOTO 600
330 A=A+2:B=B+2:C=C-2:D=D-2
340 IF F>600 THEN A=A+1:B=B+1:C=C-1:D=D-1
350 IF W=1 THEN GOSUB 440
360 IF X=1 THEN GOSUB 500
370 IF Y=1 THEN GOSUB 540
380 IF Z=1 THEN GOSUB 570
390 IF A>180 THEN GOSUB 440
400 IF B>62 THEN GOSUB 500
410 IF C<2 THEN GOSUB 540
420 IF D<2 THEN GOSUB 570
430 GOTO 150
440 W=1:A=A-4
450 REM E-5 INVERTS THE GRAPHIC LINES
460 IF W=1 THEN E=3
470 IF A<7 THEN W=0
480 IF W=0 THEN E=5
490 RETURN
500 X=1:B=B-4
510 IF X=1 THEN E=5
520 IF B<6 THEN X=0
530 RETURN
540 Y=1:C=C+4
550 IF C>175 THEN Y=0
560 RETURN
570 Z=1:D=D+4
580 IF D>59 THEN Z=0
590 RETURN
600 A=A+2:B=B+2:C=C-2:D=D-2
610 E=3
620 GOTO 350
630 IF F=1000 THEN F=0
640 PRINT CHR$(12)
650 RETURN
660 IF F<705 THEN GOTO 310
670 RETURN
```

READY

Dear Don;

First I want to thank you for publishing the 68'MICRO JOURNAL. I find it is the best computer magazine available. I recently ordered a RADIO SHACK COLOR COMPUTER and find the COLOR COMPUTER USER NOTES to be most helpful. Keep up the good work.

I'm disapointed that there are so few articles on the CT-82 terminal in any of the computer magazines. I purchased my CT-82 through you a couple of years ago and after making a few modifications I am quite satisfied with it, although it seems to be very slow in the graphics mode.I contacted SWTP some time ago when I was having trouble interfacing the CT-82 to a computer through a modem and they came up with a fix which consisted of replacing the original A-1 ROM with a new B-1 ROM. This solved my problem and improved the CT-82 operation. I wonder how many CT-82 users know about this improvement?

I've been watching for some good CT-82 graphics programs to show up but to no avail so I started experimenting the other day and came up with what I think is a very interesting DMO. I call it VARPLOT because the results can be varied so much by making slight changes in the variables. I run the program at 9600 BAUD but if the HIGH BAUD RATE OPERATION mod. detailed on page L-1 of the CT-82 USERS GUIDE is not done a baud of around 600 may be necessary because of the slow graphic response of the CT-82. Incidentally this program will only run on a CT-82 terminal.

I'll pass on the following for what it's worth. I sent in a M/O to PROGRAMMA INTERNATIONAL in BURBANK on September 28/81 for a PIE TEXT EDITOR and after waiting for 6 weeks I heard a rumor that they had gone into receivership. I tried to phone them but to no avail and wrote them again on November 24/81. I have heard nothing from them to date. I now have our local BETTER BUSINESS BUREAU working on it.

Thanks again Don. I'm going to try and make the COMPUTER FAIRE on March 19-21 so may see you in San Francisco.

```
 10 REM ******* VARPLOT *******
 20 REM **************************
 30 REM USE WITH A CT-82 TERMINAL
 40 REM LINE 50 ELIMINATES PAGING
 50 EXEC."TTYSET DP=0"
 60 REM ENTER GRAPHICS MODE
 70 PRINT CHR$(29);CHR$(22)
 80 REM DISABLE SCROLL
 90 PRINT CHR$(30);CHR$(24)
100 REM INITIAL X/Y COORDINATES
110 A=2:B=2:C=180:D=62:E=3
120 REM TURN CURSOR OFF
130 PRINT CHR$(05)
140 REM SET GRAPHICS LINE MODE
150 PRINT CHR$(29);CHR$(E);
160 PRINT CHR$(A);CHR$(B);CHR$(C);CHR$(B)
170 PRINT CHR$(29);CHR$(E);
180 PRINT CHR$(A);CHR$(B);CHR$(A);CHR$(D)
190 PRINT CHR$(29);CHR$(E);
200 PRINT CHR$(A);CHR$(D);CHR$(C);CHR$(D)
210 PRINT CHR$(29);CHR$(E);
220 PRINT CHR$(C);CHR$(B);CHR$(C);CHR$(D)
230 T=T+1:F=F+1
240 IF F<5 THEN GOTO 310
```

FLEX DISK NAME CHANGER

Sometimes it would be nice to be able to change the header information on a disk, like the disk name, number, or date without having to reformat the entire disk. Since I use both sides of my disks (flippy) but do not have double sided drives, I often do not know at the time the disk is formatted with Newdisk, what will eventually be written on the back side of a disk.

This program will allow changing the name, number, or creation date that are carried in the System Information Record (SIR) on track 0, sector 3 of a Flex disk.

Although the code listed is for the 6800, it could be assembled for the 6809 by changing the subroutine calls to the addresses for Flex-9. The program works on Flex-9 disks even though it is running on a 6800, since the SIR arrangement is the same as in Flex 2.8. It also works on 8" as well as 5" disks.

The program is called as a Flex command, with a parameter following the name which specifies the drive number. Drive zero is not permitted, although this could be changed if desired. The program is self prompting. A carriage return in answer to a question will retain the old data. A sample run follows:

```
+++ SIRFIX,1

NAME WAS XXYYZZ (RETURN) IF OK, NEW STRING IF NOT
MICRO68
DISK # WAS      1 (RETURN) IF OK, NEW STRING IF NOT

DATE WAS    11/   3/  82 (RETURN) IF OK, NEW STRING IF NOT

TASK COMPLETE

+++
```

Ralph Moslund
9273 28th Ave. NW
Seattle, WA 98117
206 784 9424

```
                    NAM    SIRFIX
                    OPT    PAG,NOG
                *DISK 9119
                *
                *REVISE DATA IN SYSTEM INFORMATION RECORD
                *
                *RUNS IN 6800-FLEX 2
                *WORKS FOR EITHER 8" OR 5" DISKS, EITHER
                *DENSITY, AND ON FLEX-9 DISKS AS WELL.
                *
                *    TRANSIENT FLEX CALL:
                *         SIRFIX,(DRIVE NUM)
                *DRIVE ZERO IS RESERVED FOR SYSTEM DISK
                *
                *
                *SIR OFFSETS
                *
0018            NAMST   EQU    $18
001A            NAMEND  EQU    $1A
0018            DISKNO  EQU    $1B
001D            FREEST  EQU    $1D
```

```
0023 001F           FREEND  EQU   $1F
0024 0022           SECRMN  EQU   $22       SECTORS REMAINING
0025 0023           DATE    EQU   $23
0026 0026           MAXTRK  EQU   $26
0027 0027           MAXSEC  EQU   $27
0028                *
0029 B4B6           FMS     EQU   $B4B6
0030 B403           CLOSE   EQU   $B403
0031 AD18           OUTCH   EQU   $AD18
0032 AD1E           PDATA   EQU   $AD1E
0033 AD24           PCRLF   EQU   $AD24
0034                *FOLLOWING PRINTS LINE WITHOUT C-R/L-F
0035 F027           PDATA1  EQU   $F027     (OR $E97E IN MIKBUG)
0036 AC14           BUFPT   EQU   $AC14
0037 000A           LNAM    EQU   NAMEND-NAMST LENGTH OF DISK NAME
0038 AD18           INBUF   EQU   $AD18
0039 AD3F           RPTERR  EQU   $AD3F
0040 AD03           WARM    EQU   $AD03
0041 AD27           NXTCH   EQU   $AD27
0042 AD39           OUTDEC  EQU   $AD39
0043 AD42           GETHEX  EQU   $AD42
0044 AD40           INDEC   EQU   $AD40
0045                *
0046 A840           FCB     EQU   $A840
0047 A800           BUF     EQU   FCB+$40
0048 A87F           STACK   EQU   $A87F
0049                *
0050                *
0051 A100                   ORG   $A100
0052 A100 7E A1 77          JMP   START
0053 A103 4E        NAMSG   FCC   'NAME WAS '
0054 A10C 04                FCB   4
0055 A10D 20        OKMSG   FCC   ' (RETURN) IF OK, NEW STRING IF NOT'
0056 A12F 0A                FCB   $A,$D,4
0057 A132 44        NUMSG   FCC   'DISK # WAS '
0058 A13D 04                FCB   4
0059 A13E 44        TYPMSG  FCC   'DISK TYPE WAS '
0060 A14C 04                FCB   4
0061 A14D 44        DATMSG  FCC   'DATE WAS '
0062 A156 04                FCB   4
0063 A157 54        ENMSG   FCC   'TASK COMPLETE'
0064 A164 0A                FCB   $A,$D,4
0065 A167 42        DRERR   FCC   'BAD DRIVE #'
0066 A172 04                FCB   4
0067                *
0068 A173 00 00     XTEMP   FDB   0
0069 A175 00        XHI     FCB   0
0070 A176 00        XLOW    FCB   0
0071                *
0072                *MAIN PROGRAM
0073                *
0074 A177 8E A8 7F  START   LDS   #STACK
0075 A17A BD AD 42          JSR   GETHEX    GET UNIT #
0076 A17D 24 09             BCC   SUNIT
0077 A17F CE A1 67  ERRDR   LDX   #DRERR    PRINT ERROR MSG
0078 A182 BD AD 1E          JSR   PDATA
0079 A185 7E AD 03          JMP   WARM
0080 A188 FF A1 75  SUNIT   STX   XHI       MOVE UNIT # TO 'A'
0081 A18B B6 A1 76          LDA A XLOW
0082 A18E 27 EF             BEQ   ERRDR     UNIT 0 IS A NO-NO
0083 A190 81 03             CMP A #3        UNIT 3 IS LIMIT
0084 A192 22 E8             BHI   ERRDR
0085 A194 CE A8 40          LDX   #FCB
0086 A197 A7 03             STA A 3,X
0087 A199 86 00             LDA A #0        TRACK
0088 A19B A7 1E             STA A 30,X
0089 A19D 86 03             LDA A #3        SECTOR
0090 A19F A7 1F             STA A 31,X
0091 A1A1 86 09             LDA A #9        FUNC CODE
0092 A1A3 A7 00             STA A X
0093 A1A5 BD A2 72          JSR   LFMS      GET THE DIR SECTOR
0094                *PRINT OLD NAME
0095 A1A8 CE A1 03          LDX   #NAMSG
0096 A1AB BD AD 1E          JSR   PDATA
0097 A1AE CE A8 90          LDX   #BUF+NAMST
0098 A1B1 C6 0A             LDA B #LNAM
0099 A1B3 BD A2 37          JSR   PSTRNG
0100 A1B6 BD A2 41          JSR   OKORNT    CHECK ANSWER
0101 A1B9 27 08             SEQ   DISKNM
0102                *A CHANGE HAS BEEN REQUESTED
0103 A1BB CE A8 90          LDX   #BUF+NAMST
0104 A1BE C6 0A             LDA B #LNAM
0105 A1C0 BD A2 52          JSR   TFRST
0106                *CHECK DISK NUMBER
0107 A1C3 CE A1 32  DISKNM  LDX   #NUMSG
0108 A1C6 BD AD 1E          JS    PDATA
0109 A1C9 CE A8 9B          LDX   #BUF+DISKNO
0110 A1CC C6 01             LDA B #1        SET FOR NO LEAD ZERO
0111 A1CE BD AD 39          JSR   OUTDEC
0112 A1D1 BD A2 41          JSR   OKORNT
0113 A1D4 27 0D             BEQ   DATECK
0114 A1D6 BD AD 40          JSR   INDEC     GET DEC # FOR DISK #
0115 A1D9 25 05             BCS   ERRHX
0116 A1DB FF A8 9B          STX   BUF+DISKNO
0117 A1DE 20 03             BRA   DATECK
0118 A1E0 7E AD 03  ERRHX   JMP   WARM
0119                *
0120                *CHECK DATE
0121                *
0122 A1E3 CE A1 4D  DATECX  LDX   #DATMSG   PRINT REQUEST
0123 A1E6 BD AD 1E          JSR   PDATA
0124 A1E9 CE A8 A3          LDX   #BUF+DATE
0125 A1EC FF A1 73          STX   XTEMP
0126 A1EF BD A2 84          JSR   PDATE     PRINT MONTH
0127 A1F2 BD A2 84          JSR   PDATE     PRINT DAY
0128 A1F5 FE A1 73          LDX   XTEMP
0129 A1F8 A6 00             LDA A X         GET YEAR IN BIN
0130 A1FA B7 A1 76          STA A XLOW
0131 A1FD 7F A1 75          CLR   XHI
0132 A200 CE A1 75          LDX   #XHI
0133 A203 C6 01             LDA B #1
0134 A205 BD AD 39          JSR   OUTDEC    PRINT YEAR
0135 A208 BD A2 41          JSR   OKORNT    SEE IF CHANGE REQUESTED
0136 A20B 27 0F             BEQ   FINIS     NO CHANGE
0137 A20D CE A8 A3          LDX   #BUF+DATE ELSE CHANGE DATE
0138 A210 FF A1 73          STX   XTEMP
0139 A213 BD A2 A0          JSR   SDATE     STORE MONTH
0140 A216 BD A2 A0          JSR   SDATE     STORE DAY
0141 A219 BD A2 A0          JSR   SDATE     STORE YEAR
0142                *
0143                *DON , SO WRITE SECTOR BACK
0144                *
0145 A21C CE A8 40  FINIS   LDX   #FCB
0146 A21F 86 0A             LDA A #10       WRITE SECTOR
0147 A221 A7 00             STA A X
0148 A223 86 03             LDA A #3        RESET SECTOR NO
0149 A225 A7 1F             STA A 31,X
0150 A227 86 00             LDA A #0
0151 A229 A7 1E             STA A 30,X
0152 A22B BD A2 72          JSR   LFMS
0153 A22E CE A1 57          LDX   #ENMSG
0154 A231 BD AD 1E          JSR   PDATA
0155 A234 7E AD 03          JMP   WARM
0156                *
0157                *SUBROUTINE SECTION
0158                *
0159                *PRINT STRING '#' CHAR LONG
0160                *POINTED TO BY X
0161                *
0162 A237 A6 00     PSTRNG  LDA A X
0163 A239 BD AD 18          JSR   OUTCH
0164 A23C 08                INX
0165 A23D 5A                DEC B
0166 A23E 26 F7             BNE   PSTRNG
0167 A240 39                RTS
0168                *
0169                *SEE IF OLD DATA IS STILL OK
0170                *RETURN 'NE' IF NOT WITH NEW DATA IN BUFFER
0171                *
0172 A241 CE A1 BD  OKORNT  LDX   #OKMSG
0173 A244 BD F0 27          JSR   PDATA1
0174 A247 BD AD 18          JSR   INBUF
0175 A24A FE AC 14          LDX   BUFPT     GET BUFFER POINTER
0176 A24D A6 00             LDA A X         SEE WHAT FIRST CHAR IS
0177 A24F 81 0D             CMP A #$D       IS IT C-R
0178 A251 39                RTS
0179                *
0180                *TRANSFER A STRING UP TO C-R
0181                *
0182 A2 2 FF A1 73  TFRST   STX   XTEMP
0183 A255 37                PSH B
0184 A256 BD A2 6B          JSR   NULLS     FILL SPACE WITH NULLS FI
0185 A259 33                PUL B
0186 A25A FE A1 73          LDX   XTEMP
0187 A 5D BD AD 27  TFRST2  JSR   NXTCH
0188 A260 81 0D             CMP A #$D
0189 A262 27 06             BEQ   OUTEX
0190 A264 A7 00             STA A X
0191 A266 08                INX
0192 A267 5A                DEC B
0193 A268 26 F3             BNE   TFRST2    GO IF NOT AT LIMIT
0194 A26A 39        OUTEX   RTS
0195                *
0196                *FILL '#' CHAR WITH N LLS STARTING AT X
0197                *
0198 A26B 6F 00     NULLS   CLR   X
0199 A26D 08                INX
0200 A26E 5A                DEC B
0201 A26F 26 FA             BNE   NULLS
0202 A271 39                RTS
0203                *
0204                *LOCAL FMS CALL
0205                *
0206 A272 BD B4 B6  LFMS    JSR   FMS
0207 A275 26 01             BNE   LFMS2
0208 A277 39                RTS
0209 A278 8E A8 7F  LFMS2   LDS   #STACK
0210 A27B BD AD 3F          JSR   RPTERR
0211 A27E BD B4 03          JSR   CLOSE
0212 A281 7E AD 03          JMP   WARM
0213                *
0214                *PRINT DIGIT IN DATE
0215                *
0216 A284 FE A1 73  PDATE   LDX   XTEMP
0217 A 87 A6 00             LDA A X         GET BINARY #
0218 A289 08                INX             STEP TO NEXT BYTE IN DAT
0219 A28A FF A1 73          STX   XTEMP
0220 A28D B7 A1 76          STA A XLOW      MEMORY WORKSPACE
0221 A290 7F A1 75          CLR   XHI       WIPE OUT MSP
0222 A293 CE A1 75          LDX   #XHI
0223 A296 C6 01             LDA B #1        SUPPRESS LEAD ZERO
0224 A298 BD AD 39          JSR   OUTD C
0225 A29B 86 2F             LDA A #$2F      PRINT SLASH
0226 A29D 7E AD 18          JMP   OUTCH
0227                *
0228                *STORE DATE
0229                *
0230 A2A0 BD AD 40  SDATE   JSR   INDEC     GET DECIMAL DIGITS
0231 A2A3 24 03             BCC   SDATE2
0232 A2A5 7E AD 03          JMP   WARM
0233 A2A8 FF A1 75  SDATE2  STX   XHI       STD E 16 BITS
0234 A2AB FE A1 73          LDX   XTEMP
0235 A2AE B6 A1 76          LDA A XLOW      EXTRACT ONLY LO BYTE
2 6 A201 A7 00              STA A X         STORE IN BIN
0237 A203 08                INX
0238 A204 FF A1 73          STX   XTEMP
0239 A207 39                RTS
0240                *
0241                          END   $A100

NO ERROR(S) DETECTED
```

UTILITY SUMMARY

Here is a program which will pour out pages of text to the screen without needing a second file like 'LIST.CMD'. It has the additional feature of stopping the output at the end of each page and prompting for a space to continue or carriage return to abort to FLEX.

The text accompanying the program is a ready reference for the FLEX utilities, designed to be a (human) memory prompt for the syntax and use of some of the less commonly used utilities without having to consult the FLEX manual.

The text can of course be anything you wish.

To use, type **++UTILITY

Aims - command file to display pages of text to the screen
- stop at page boundaries
- change text without reformatting pages manually
- easy as possible to enter text
- optional header and trailer lines at top and bottom of screen
- optional force end of page
- demonstrate use of macros

Techniques - uses a macro for the insertion of CR, LF, EOT
- text is a parameter of the macro
- second parameter (+) for putting more text on second line of source to appear on same line of output.
- second parameter (PAG) for forcing end of page
- Operates under FLEX (uses standard FLEX I/O)

Uses - as here, ready reference for utilities and their parameters
- any application requiring pages of text to be displayed on the screen

System requirements - 6809 running FLEX
- TSC's ASM.CMD macro assembler (the later, correct, 40 sector version)

Availability - I will provide a copy on any 5" disks which reach me. Disks are expensive here, so I would have to charge $10 if no disk is supplied. At 29 sectors, there is a considerable amount of typing.
The source is presented here unassembled as this is easier to read than the expanded macro output.

John Caldwell
31 Harris St
Marton
New Zealand

```
UTILITY     PARAMETER(S)                COMMENTS

APPEND      FILE1 FILE2 .. FILETOTAL    concatenate two or more files
ASMB        FILE                        TSC assembler
ASM09       FILE                        GIMIPC assembler, requires ASM09.CMD
AR          <input drive> <output drive> archive - copy only if more recent
ASN         W=<drive> S=<drive>         assign system and working drives
BUILD       FILE                        build a text file  . to exit
CAT         match list                  catalogue of files on a disk
COPY        FILE drive or match list    copy files disk to disk
CAMAT                                   format calcomp hard disk
CLEAN       drive                       8" disk cleaner
DATE        <mm,dd,yy>                  set current date
DELETE      FILE                        delete files
ECHO        <string>                    messages for EXEC files
ED          FILE                        TSC editor
EXEC        FILE                        execute a text file as commands
FILESORT                                sort files on various disks - prompting
```

```
   "SPACE" to next page, "RETURN" to abort
UTILITY     PARAMETER(S)                COMMENTS

FIX         FILE                        modify binary files on disk
            $ AAAA AAAA   add block of addresses
            E             exit
            L             display file limits
            M AAAA        memory examine and change
            N             next line
            P AAAA        peek at file
            T AAAA        specify transfer address
            V AAAA AAAA   view section of file
            X             abort without changing file
HEXCH       <hex string>                like ECHO but hex values
I           <command>                   input chars from disk not terminal
JUMP        <hex address>               jump to program at address
LINK        FILE (normally FLEX)        sets up bootstrap loader
LIST        FILE <line range> (^NP)     list text file (numbers, page)
MAP         FILE                        file load and transfer addresses
MEMEND      (?) or (value)              end of memory examine or set
MIRROR      <drive> <drive>             mirror image copy of a disk
MV          FILE1 FILE2                 move files - can rename, delete
NEWDISK     <drive>                     format a disk
```

```
   "SPACE" to next page, "RETURN" to abort
UTILITY     PARAMETER(S)                COMMENTS

N           <command>                   answers NO - used by EXEC files
O           FILE <command>              routes output to disk
P           <command>                   routes output to printer
PDEL        <drive> <match list>        prompting match list delete
PSP         <spool file> <print tod> ^RPT printer spooler
PROT        FILE < D W C I >            delete write cat or remove protection
PUTBOOT                                 put bootstrap onto disk
Q           <command>                   like "P" for Qume printer
```

```
        OPT     PAG
        PAG
*******************************************************
*
* PROGRAM TO DISPLAY READY REFERENCE OF DOS UTILITIES
*
* USE IN THE FORM
* **UTILITY
*
*                       WRITTEN BY    J D CALDWELL
*                       LAST MOD      13 JAN 83
*
*******************************************************
*
* Position independent code
*
* Extra text may be added anywhere in the file, with
* line delimiters using the macro LINE. This puts in
* C/R, L/F each line and an EOT at the end of each
* page. The total number of pages is taken care of by
* the assembler.   ( ASM of course )
* Macro parameter  +  suppresses CR, LF on the current
*    source line.
* Macro parameter  PAG  adds extra linefeeds to force
*    a page boundary.
* Assumes 24 line screen. Modify at ** in macro.
*
*******************************************************
*
LINENO SET     0
PAGNO  SET     0
*
LINE    MACRO
        FCC     /&1/    text handled here
        IFNC    &2,+    more text on next source line
        FCB     $0D,$0A
        IFC     &2,PAG  force page end
        OUP     16-LINENO    ** 24 line page
        FCB     $0A
        ENDC
LINENO SET     19           ** 24 line page
        ENDIF
LINENO SET     LINENO+1
        IF      LINENO>20    ** 24 line page
        FCB     $04
LINENO SET     0
PAGNO  SET     PAGNO+1
        ENDIF
        ENDIF
        ENDM
*
*******************************************************
*
        ORG     $0100
*
WARMS   EQU     $CD03
GETCHR  EQU     $CD15
PSTRNG  EQU     $CD1E
*
*******************************************************
*
UTLIST  BRA     START
VN      FCB     1
PAGE    FDB     MSG3-*      offset to text
PAGTOT  FCB     PAGCNT
*
START   LEAX    PAGE,PCR
        LDD     ,X
        LEAX    D,X
        STX     PAGE,PCR    to make code relocatable
LOOP    LEAX    MSG1,PCR
        JSR     PSTRNG      print top heading
        LDX     PAGE,PCR
        JSR     PSTRNG      print page of text
        LEAX    1,X
        STX     PAGE,PCR
        LEAX    MSG2,PCR
        JSR     PSTRNG      print bottom heading
        JSR     GETCHR
        CMPA    #$0D
        BEQ     ABORT
        DEC     PAGTOT,PCR
        BNE     LOOP
*
ABORT   JMP     WARMS
*
*******************************************************
*  Header and trailer messages
*
MSG1    FCC     'UTILITY     PARAMETER(S)
        FCB     $0D,$0A,$04
*
MSG2    FCC     '   "SPACE" to next page, "RETURN" to abort'
        FCB     $0D,$04
*
*******************************************************
*  Bulk of text
*
MSG3    EQU     *
 LINE 'APPEND     FILE1 FILE2 .. FILETOTAL      '.+
 LINE 'concatenate two or more files'

 LINE 'ASMB       FILE                          '.+
 LINE 'TSC assembler'

 LINE 'ASM09      FILE                          '.+
 LINE 'GIMIPC assembler, requires ASM09.CMD'

 LINE 'AR         <input drive> <output drive> '.+
 LINE 'archive - copy only if more recent'

 LINE 'ASN        W=<drive> S=<drive>           '.+
 LINE 'assign system and working drives'

 LINE 'BUILD      FILE                          '.+
 LINE 'build a text file  . to exit'

 LINE 'CAT        match list                    '.+
 LINE 'catalogue of files on a disk'

 LINE 'COPY       FILE drive or match list      '.+
 LINE 'copy files disk to disk'

 LINE 'CAMAT                                    '.+
 LINE 'format calcomp hard disk'

 LINE 'CLEAN      drive                         '.+
 LINE '8" disk cleaner'

 LINE 'DATE       <mm,dd,yy>                    '.+
 LINE 'set current date'

 LINE 'DELETE     FILE                          '.+
 LINE 'delete files'

 LINE 'ECHO       <string>                      '.+
 LINE 'messages for EXEC files'

 LINE 'ED         FILE                          '.+
 LINE 'TSC editor'

 LINE 'EXEC       FILE                          '.+
 LINE 'execute a text file as commands'

 LINE 'FILESORT                                 '.+
 LINE 'sort files on various disks - prompting',PAG

 LINE 'FIX        FILE                          '.+
 LINE 'modify binary files on disk'
```

```
LINE '           B AAAA AAAA  add block of addresses'
LINE '           E            exit'
LINE '           L            display file limits'
LINE '           M AAAA       memory examine and change'
LINE '           N            next line'
LINE '           P AAAA       peek at file'
LINE '           T AAAA       specify transfer address'
LINE '           V AAAA AAAA  view section of file'
LINE '           X            abort without changing file'

LINE 'HEXMD      <hex string>                    ',+
LINE 'like ECHO but hex values'

LINE 'I          <command>                       ',+
LINE 'input chars from disk not terminal'

LINE 'JUMP       <hex address>                   ',+
LINE 'jump to program at address'

LINE 'LINK       FILE (normally FILE)            ',+
LINE 'sets up bootstrap loader'

LINE 'LIST       FILE <line range> (+NP)         ',+
LINE 'list text file (numbers, page)'

LINE 'MAP        FILE                            ',+
LINE 'file load and transfer addresses'

LINE 'MEMEND     <?> or <value>                  ',+
LINE 'end of memory examine or set'

LINE 'MIRROR     <drive> <drive>                 ',+
LINE 'mirror image copy of a disk'

LINE 'MV         FILE1 FILE2                     ',+
LINE 'move files - can rename, delete'

LINE 'NEWDISK    <drive>                         ',+
LINE 'format a disk'

LINE 'N          <command>                       ',+
LINE 'answers NO - used by EXEC files'

LINE 'O          FILE <command>                  ',+
LINE 'routes output to disk'

LINE 'P          <command>                       ',+
LINE 'routes output to printer'

LINE 'PDEL       <drive> <match list>            ',+
LINE 'prompting match list delete'

LINE 'PSP        <spool file> <prnt cmd> sRPI    ',+
LINE 'printer spooler'

LINE 'PROT       FILE < D W C X >                ',+
LINE 'delete write cat or remove protection'

LINE 'PUTBOOT                                    ',+
LINE 'put bootstrap onto disk'

LINE 'Q          <command>                       ',+
LINE 'like "P" for Qume printer',PAG

LINE 'QCHECK     C/R restart printing            ',+
LINE 'spooler queue check'

LINE '           Q      print queue contents     ',+
LINE '     commands prompted for later'

LINE '           R #N X repeat file N, X times'

LINE '           D #N   delete file N from queue'

LINE '           T      terminate current job'

LINE '           N #N   makes next file printed #N'

LINE '           B      printing stops at end of job'

LINE '           G      restart after B'

LINE '           K      kill printer queue'

LINE 'RENAME     FILE1 FILE2                     ',+
LINE 'change name of a file'

LINE 'RM         <?> or <size>                   ',+
LINE 'reserve memory for printer drivers'

LINE 'RPROMPGM   FILE                            ',+
LINE '2716 reader with SWTPC hardware'

LINE 'RUN        </><load addr> <command>        ',+
LINE 'load & opt execute posn indep code'

LINE 'S          <command>                       ',+
LINE 'like "P" for serial printer'

LINE 'SAVE       FILE <begin> <end> <trans>      ',+
LINE 'save memory to disk'

LINE 'SBOX       <param>=<value>                 ',+
LINE 'configure memory and I/O'

LINE 'SDC        FILE1,FILE2...                  ',+
LINE 'single disk copy for up to 5 files'

LINE 'SP         <command>                       ',+
LINE 'like "P" for IBM typewriter'

LINE 'STARTUP                                    ',+
LINE 'file of cmds executed at powerup'

LINE 'SPLIT      <input> <out1> <out2> <N>       ',+
LINE 'split file at line N'

LINE 'SUM        FILE                            ',+
LINE 'checksum of file'

LINE 'TIME       <command>                       ',+
LINE 'cmnd execution time - uses RTC',PAG

LINE 'TTYSET     BS=hh backspace                 ',+
LINE 'set terminal characteristics'

LINE '           BE=hh backspace echo            ',+
LINE 'parameters on cmnd line'

LINE '           DL=hh delete character          '

LINE '           EL=hh end of line'

LINE '           OP=hh depth count'

LINE '           WD=dd width'

LINE '           NL=dd null count'

LINE '           TB=hh tab character'

LINE '           EJ=dd eject count'

LINE '           PS=Y/N pause control'

LINE '           ES=hh escape character'

LINE 'TOUCH      FILE                            ',+
LINE 'change date in dir. to current'

LINE 'USENF                                      ',+
LINE 'for mini floppy on B= system'

LINE 'UCAL                                       ',+
LINE 'for calcomp fixed disk'

LINE 'VER        FILE                            ',+
LINE 'display version of utility'

LINE 'VERIFY     <ON> or <OFF>                   ',+
LINE 'read back from disk after write'

LINE 'VOLSET     <disk name> <number>            ',+
LINE 'change disk volume name'

LINE 'WRITPROM   FILE <addr>                     ',+
LINE 'SWTPC hardware EPROM programmer'

LINE 'XOUT       <drive>                         ',+
LINE 'delete files with .OUT extension'

LINE 'Y          <command>                       ',+
LINE 'answer Y in EXEC files'

LINE 'ZAP        <match list>                    ',+
LINE 'delete files without prompting',PAG
*
        FCB     $04
*
PAGCNT  EQU     PAGNO
        END     UTLIST
```

WINDRUSH PL9 UPDATE

A few days ago, I received a package from Windrush. It contained an update on their PL9 package that I reviewed in the January '68' Micro Journal. About then, the January issue arrived at Windrush and they sent me a nice letter telling me what I had found out by looking through their update package. A quick rundown on the new features is in order.

1. They have implemented the 32 bit REAL data type as an integral part of PL9. That is, you may declare variables of type REAL, arrays of REAL, you may use a REAL variable in an equation, etc. The compiler handles mixed mode arithmetic (that is, it "promotes" INTEGER variables used in calculations with REAL variables, so that the result is a REAL. It has the necessary commands to convert between data types. There are new Library packages that allow input and output of REAL numbers, etc.

2. I had complained about the lack of any example programs. Now the manual contains listings for a pair of demonstration programs as well as the source for the Library files.

3. Library modules may be "trimmed" by making versions of them with only the procedures that your program will use. I had just finished translating a Pascal program of four pages or so to PL9, and I had done just that, when the response from Windrush arrived. Such "trimming" did reduce the size of the output code significantly.

4. The revised manual is considerably better than the one originally supplied.

I ought to say something about manuals here. I am usually very critical of a manual if I can't figure out how to use some feature of some new software by reading the manual.

I have said previously, that I think it is impossible for the author of a piece of software to write the manual

for his own product. Authors are so familiar with their own work that they inevitably leave something out. Though many do considerably better than I do with my software, everyone seems to leave out a detail or two, or present it in an ambiguous manner.
Windrush, even in the original manual did no worse than most of the software houses do.

I think that if an author must write the manual for his own software, it should be tested by giving it to someone totally unfamiliar with that software, and that person should be turned loose at the computer to try to run the software, using only the instruction manual. Any questions that come up can be noted, and the answers included in the manual. I'm not proposing that someone be brought in off the street. A reasonably facile programmer or computer user may be used, provided he is unfamiliar with the particular software in question.

Back to the particulars of Windrush PL9. Seeing that the REALS had been itegrated into PL9, I decided to translate a four page program from Pascal, to see if I would run into any difficulties. Other than having to dig once or twice into the source listings for the Library files to see just what parameters were expected by the ASCII to BINARY conversion procedure for REAL numbers, I had no problems at all.

The conversion from Pascal to PL9 was straightforward. Within the code itself, there were virtually no changes. WHILE <CONDITION> DO in Pascal becomes WHILE <CONDITION> in PL9. Procedure declarations and local variable declarations are almost trivially different in syntax. X,Y : INTEGER; in Pascal becomes INTEGER X,Y ; in PL9. Procedures in PL9 are more like procedures in "C" than in Pascal. In both PL9 and "C", a procedure can be like a Pascal procedure, or like a Pascal FUNCTION. That is, a procedure may return a value as a Pascal Function does, or it may not, at the programmer's option.

After programming a few pages of PL9, I was able to write a formatted output procedure for the REAL and INTEGER variables, that duplicates that feature of Pascal, that is:

```
WRITE (INTVAR :7);  § PASCAL †
PRINTI (INTVAR, 7); /* PL9 */


OR
WRITE (REALVAR 12:3);
PRINTR (REALVAR, 12,3);
```

Unfortunately, since the print procedure must be passed a REAL value in one case and an INTEGER in the other, two separate print procedures are required for numbers.

With these added procedures, I was able to get my translated program running as well as the original, in less than a day. I now feel relatively comfortable with PL9, and will try to use it in some of my programming in the near future.

What I learned to appreciate the most, was the very short compile time of PL9. One most useful feature is the fact that the compiler will indicate an error, and return to the editor pointing at the line in question. Generally, my errors were obvious typos or syntax errors that were correctable with the editor. The program can then be recompiled instantly. My four page program with three libraries included, compiled in 32 seconds, not including the load time for PL9, about 8 seconds. PL9 only has to be loaded once, and the program can be recompiled indefinitely without reloading it. Though I still like my screen editor for entering the program source the first time, I found the co-resident editor to be of great use in fixing my dumb errors.

This is a very good product, and it has been improved impressively in a relatively short time.

Ron Anderson

CLASSIFIED ADVERTISING

HELP

Are there any FLEX Users Groups that I can exchange software with? You're magazine has a lot of good programs, but I have missed most issues and finding back issues could get expensive.
Frank J. Wilson, 2468 Elden St. Apt J, Costa Mesa Ca. 92627.

CLASSIFIEDS

SOFTWARE WIZARDS WANTED
EXPERT C AND ASSEMBLY LANGUAGE PROGRAMMERS TO PARTICIPATE IN CHALLENGING 6809 AND 68000 SOFTWARE PROJECTS.
MAIL RESUME AND SALARY REQUIREMENTS TO: MICROWARE SYSTEMS CORPORATION, 5835 GRAND AVENUE, DES MOINES, IOWA 50312, ATTN: JEANNE KAPLAN. NO PHONE CALLS PLEASE.

***

SS-50C SYSTEM (REPOPULATED, UNDERBUGGED): SSB 6809 CPU, F A D MOTHERBOARD, 2-PORT SERIAL I/O, SWTP 4K RAM, POWER SUPPLY, DOCUMENTATION. FIRST REASONABLE OFFER. STEVE ELKIND, 6340 MARCHAND, PITTSBURGH, PA 15206. (412)361-3788.

***

I SWITCHED TO GIMIX MINI-MINI AND WILL SELL SWTP CDS-1 HARD DISK WITH STAND FOR A FAIR PRICE. WORKS FINE.
JIM PAZARENA, PO BOX 490, QUEEN CHARLOTTE ISLANDS, BC CANADA V0T1S0

***

FOR SALE: EXCELLENT CONDITION SSA6809 SYSTEM, 56K, DCR4AFDC, 2X5001 (ADR) DRIVES, 25/2P IO, CLOCK. OSK: MULTITASKING, MPASIC09, ED, ASM, DEBUG. SCREENEDITOR AND DATABASE SOFTWARE. M19 TERMINAL W/UTILITIES. COMPLETE SYSTEM LESS PRINTER. $3800.
DOUG RAMERS, (302)474-6366, 697-3444. 245 CAMBRIDGE RD, CAMDEN, DEL 19934.

***

SWTPC AC-30 CASSETTE INTERFACE $29. MARE MP-C $5. TWO MP-LA $20 EA., MP-T $30, THREE MP-N $10 EA., F&D FHB-1 VIDEO BOARD $90, SOUTHEASTERN MICRO DM-64 64K MEM BOARD $50. ALL BOARDS ARE WORKING WITH DOCUMENTATION.
RICK RICHARDSON, 1810 ARYON CREEK RD., WINSTON, OR 97496 (503)679-6217

***

SEALS 8K MEMORY BD. $60.00, DRC 16K MEMORY BD. $90.00, GT-64 TERMINAL $190.00., GT-6144 GRAPHICS BD $40.00. DOCUMENTATION FOR ALL INCLUDED.
KURT WOLFE (317) 293-9701

***

SWTP 32 32K $150.00, TI743 $450.00, MPA2 RAM CARD $40.00, PERCOM LFD400 AND SA400 WITH CABINET,CABLE, AND SOFTWARE $225. PERCOM CIS304 CASSETTE INT $35.
TOM ADAMS, 11608 BROAD OAKS, AUSTIN, TX 78759

***

SWTPC SYSTEM: MP-A2, SMARTBUG, 48K, MP-S, GT-64, GT-VM, AC-30, SSB BFD-68 DISK CONTROL & DRIVE, TC-3 4800 BAUD TAPE INTERFACE, $1800/OFFER
JOHN (206) 878-3629, 6-10PM PST

***

TEN MOST-ASKED QUESTIONS

ABOUT DYNACALC™

THE ELECTRONIC SPREAD-SHEET FOR 6809 COMPUTERS

1. **What is an electronic spread-sheet, anyway?**
   Business people use spread-sheets to organize columns and rows of figures. DYNACALC simulates the operation of a spread-sheet without the mess of paper and pencil. Of course, corrections and changes are a snap. Changing any entered value causes the whole spread-sheet to be re-calculated based on the new constants. This means that you can play, 'what if?' to your heart's content.

2. **Is DYNACALC just for accountants, then?**
   Not at all. DYNACALC can be used for just about any type of job. Not only numbers, but alphanumeric messages can be handled. Engineers and other technical users will love DYNACALC's sixteen-digit math and built-in scientific functions. There's even a built-in sort command, so you could use DYNACALC to manage small data bases - up to 256 records.

3. **What will DYNACALC do for ME?**
   That's a good question. Basically the answer is that DYNACALC will let your computer do just about anything you can imagine. Ask your friends who have VisiCalc, or a similar program, just how useful an electronic spread-sheet program can be for all types of household, business, engineering, and scientific applications.

4. **Do I have to learn computer programming?**
   NO! DYNACALC is designed to be used by non-programmers, but even a Ph.D. in Computer Science can understand it. Built-in HELP messages are provided for quick reference to operating instructions.

5. **Do I have to modify my system to use DYNACALC?**
   Nope. DYNACALC uses any standard 6809 configuration, so you don't have to spend money on another CPU board or waste time learning another operating system.

6. **Will DYNACALC read my existing data files?**
   You bet! DYNACALC has a beautifully simple method of reading and writing data files, so you can communicate both ways with other programs on your system, such as the Text Editor, Text Processor, Sort/Merge, RMS data base system, or other programs written in BASIC, C, PASCAL, FORTRAN, and so on.

7. **How fast is DYNACALC?**
   Very. Except for a few seldom-used commands, DYNACALC is memory-resident, so there is little disk I/O to slow things down. The whole data array (worksheet) is in memory, so access to any point is instantaneous. DYNACALC is 100% 6809 machine code for blistering speed.

8. **Is there a version of DYNACALC for MY system?**
   Probably. You need a 6809 computer (32k minimum) with FLEX or UniFLEX operating system. A version for OS-9 is also in the works. You also need a decent CRT terminal, one with at least 80 characters per line, and direct cursor addressing. If your terminal isn't smart enough for DYNACALC, you probably need a new one anyway. The UniFLEX version of DYNACALC also allows you to mix different brands of terminal on the same system. There's also a special version of DYNACALC for Color Computers equipped with FLEX and Data-Comp's F-MATE. A version for Frank Hogg's Color Computer FLEX is also being done.

9. **How much does DYNACALC cost?**
   The FLEX versions are just $200 per copy; UniFLEX version $395. Foreign orders add $10 per copy for postage. We encourage dealers to handle DYNACALC, since it's a product that sells instantly upon demonstration. Call or write on your company letterhead for more information.

10. **Where do I order DYNACALC?**
    See your local DYNACALC dealer, or order directly from CSC at the address below. We accept telephone orders from 10 a.m. to 6 p.m., Monday through Friday. Call us at 314-576-5020. Your VISA or MasterCard is welcome. Please specify diskette size for FLEX versions. Software serial number is required for the UniFLEX version of DYNACALC.

ORDER YOUR DYNACALC™ TODAY

Foreign Dealers:

Australia & Southeast Asia: order from Paris Radio Electronics, 7A Burton St., Darlinghurst, NSW 2010 Sydney. Telephone: 02-357-5111.

United Kingdom: order from Compusense, Ltd., PO Box 169, London N13 4HT. Telephone: 01-882-0681.

Scandinavia: order from Swedish Electronics hk AB, Murargatan 23-25, Uppsala S-754 37 Sweden. Telephone: 18-2530-00.

Computer Systems Center
13461 Oliva Blvd.
Chesterfield, MO 63017
(314) 576-5020

UniFLEX software prices include maintenance for the first year.

DYNACALC, DYNAMITE, and DYNAMITE + are trademarks of Computer Systems Center.

F-MATE is a trademark of Data-Comp.
VisiCalc is a trademark of VisiCorp.
OS-9 is a trademark of Microware and Motorola.
FLEX and UniFLEX are trademarks of TSC.

ALSO FROM CSC

DYNAMITE +
"THE CODE BUSTER"
now available for UniFLEX OS-9 version soon

DYNAMITE + is a new version of DYNAMITE, our popular 6809/6800 disassembler package for 6809 FLEX. Present users of DYNAMITE can upgrade to DYNAMITE + by sending us the original DYNAMITE diskette and $40 (plus $5 for foreign postage). DYNAMITE + does everything DYNAMITE does, and more! A cross-reference generator has been added, label files are now maintained only in text form (LABEL EQU $xxxx), and boundary file specifications have been tremendously simplified, which makes it easier to disassemble large programs containing lots of big tables.

The UniFLEX version of DYNAMITE + does everything the FLEX version does, and also automatically handles system calls and 'Info' areas.

DYNAMITE + is available for $100 per copy on FLEX (specify diskette size), and $300 on UniFLEX. Foreign orders add $5 per copy for postage.

SCREEN EDITOR/WORD PROCESSOR FOR DOS6x, FLEX, OS-9

IF YOU ARE CONSIDERING UPGRADING YOUR EDITING AND PROCESSING CAPABILITIES, HERE ARE A FEW GOOD REASONS TO CONSIDER SCREDITOR...

[illegible] #1 - SCREEN EDITING

[illegible] editing [illegible] is the most efficient way [illegible] to enter your thoughts into the computer. Since you see the actual text you are entering on the screen as if you were looking at a [illegible] page, organization becomes a matter of observation rather than [illegible].

[illegible] #2 - [illegible] SCREEN FORMATTING

With dynamic screen formatting, the screen becomes an image of what the completed job will look like. Within the capabilities of the display terminal itself, you are constantly looking at what you would have if your job were printed. Every change you make, every addition or deletion, is [illegible] as a fully-justified, ready-to-print form. You no longer need to guess what all those cryptic processing commands you may have used with other software will turn your finished job into!

[illegible] #3 - NO MACHINE LANGUAGE PATCHING

[illegible] III is unique in that you NEVER have to make machine language patches to integrate it into your system. If you can answer a series of questions about your printer and terminal's capabilities, installation is complete.

[illegible] #4 - NO EXTRA "HIDDEN" PROGRAM COSTS

SCREDITOR III is complete as purchased. You get screen editing, formatting, mailing list merging and much more, all in a single program. And it [illegible]. Even though the competition can provide many of the features of SCREDITOR III, how much will you pay? And, more important, how many programs will you have to buy?

[illegible] #5 - OPERATING SYSTEM SUPPORT

To our knowledge, SCREDITOR III supports more processor and operating system combinations than ANY other single editing and word processing package available. [illegible] (level one and two), Flex and SSB DOS. 6800 and 6809 versions are available. Except for those commands which are peculiar to a particular operating system, SCREDITOR III is identical in operation in all versions.

[illegible] #5 1/2 - OPERATING SYSTEM SUPPORT [illegible]

If you purchase another [illegible] for your [illegible] system and decide to run [illegible], how much will it cost you for the [illegible] copy of their software (if it's available, [illegible])? With SCREDITOR III, tell us you are upgrading and you pay [illegible] of the price you paid for the first copy and you can have the same great capabilities with both operating systems!

[illegible] #6 - CAPABILITIES

[illegible] SCREDITOR III the only COMPLETE editing and word processing program available, it's also the most powerful. See if the other combinations of programs available [illegible] you true multi-column editing. Or allow you to add and decimal align rows and columns of numeric data. Or allow you to change margins and tabs at any time during editing just like a typewriter. Or give you commands to re-justify and change the width of paragraphs and lines on-screen, while you watch. Or edit and [illegible] commands. Or allow essentially unlimited mixed text and command string replacement with a single keystroke (macros). Or let you save all of those macros to a data file to be re-loaded for use in later edit sessions. Or edit your macros. Or...

[illegible] #7 - [illegible]

SCREDITOR III has the most comprehensive documentation available for any [illegible] editor/word processor. The Technical Reference Manual is currently about 100 pages long. The Training and User's Guide we are now preparing will be about another 100 pages. We're also working on our [illegible] Tutorial Cassette (two hours of instruction). We even provide a [illegible] which gives you the syntax of all of SCREDITOR III's commands and at least one [illegible] of description on each.

REASON #8 - PRICE

Considering its capabilities, SCREDITOR III is the [illegible] program of its type available today. Period.

REASON #9 - HAPPINESS

We want your business. Think how happy you will make us if you decide to buy SCREDITOR III!

The following information should answer most of your questions about SCREDITOR III. If you have additional questions, please call during our office hours, or, better still, order the manual (when you decide to buy SCREDITOR III, we'll apply your manual purchase against the software price).

PROCESSORS SUPPORTED - 6800, 6809

OPERATING SYSTEMS - FLEX, SSB DOS, OS-9 (level I and II)

DISPLAY REQUIREMENTS - Serial terminal with computer-addressable cursor. 4800 baud minimum data rate, 9600 or better preferred. Clear EOL and clear EOS not required but highly desirable. Insert and Delete line improve display speed. Suggested terminals include Televideo, Heath/Zenith H-19, SWTPCs. Virtually all addressable-cursor terminals can be used. ANSI terminals supported by the time you read this.

Memory-mapped displays require software to provide terminal look-alike capabilities. Sample software included with FLEX and DOS6x orders.

Minimum display size is 16x64. Maximum is 255x255. No software patching is required to adapt software to display size.

Unique configurations may be supported by calls to external user-supplied sub-routines.

KEYBOARD REQUIREMENTS - ASCII character set with control codes required. Function keys generating multiple-character sequences also supported. Control code and function key assignments user-defined and changeable at any time without patching.

KEYBOARD INTERFACE - PIA parallel, ACIA serial, or external keyboard handlers. Interrupt-driven operation (queue-buffered type-ahead) is a standard user-selected option.

PRINTER HARDWARE - Current version supports standard printers. Specialty printer capabilities controllable by transmission of control or escape sequences imbedded in text. Underline, boldface, double width simulated on printers with backspace capability. Proportional printing available soon.

PRINTER INTERFACE - Program uses existing user-supplied software to transfer characters from SCREDITOR III to printer hardware by calls to user sub-routines.

COMMAND SET - Fourty-six commands (or more by the time you read this) with multiple options on many commands. Thirty-one single key commands for often-used operations such as cursor movements, inserts, deletes, paragraph formatting, etc...

OPERATION AND DISPLAY - Text entry by 'window-type' screen/cursor editing. Commands by single-key or line entry. Continuous display of current tabs/margins, edit modes, cursor row and column, current line number, horizontal display range.

REQUIRED MEMORY - (program memory) - Program size - approximately 20 kilobytes. Minimum memory for useful editing - 32 kilobytes.

DISK MEMORY - Configuration files (3 minimum) - 1 to 3 sectors each
HELP file - disk-resident, approximately 19 kilobytes

** AT LAST! **

AVAILABILITY - Please allow one to two weeks for delivery. Disks are built as orders are received to allow the most current possible revision of software to be shipped.

PRICE - SCREDITOR III manual, disk $150.00
manual only (price may be applied to software order) . . . . . $ 40.00

IMPORTANT INFO

We accept Mastercharge and VISA credit card orders, personal and business checks drawn on U.S. banks and in U.S. dollars, and COD's to the continental United States. Domestic orders shipped by United Parcel Service unless otherwise requested. Overseas orders shipped by air parcel. COD's shipped by United Parcel Service only. Open accounts to government agencies, educational institutions, established accounts and Fortune 1000's only. Virginia residents add 4% state sales tax. On orders under $100, add $5 for shipping and handling. On overseas orders add $10 for shipping.

THE BEST NEWS!

In our haste to do business, too often the temptation to compromise our faith weighs heavily. Even though we have forgiveness when we confess our sins from a God who loves us, we would like to take a moment to ask forgiveness from any of our customers whom we have slighted. Orders do get lost or mis-shipped, and mistakes do [illegible]. If we fail to completely please any of you in either our responses to your order, inquiries, or for whatever reason, let us know. We will do anything in our power to satisfy your problems. We thank all of our customers for your prayers, and we pray for each of you that you will come to know the love of God in your lives that we have found!

ALFORD & ASSOCIATES
P.O. BOX 6683
RICHMOND, VA., 23230
804-320-6722

CoCo SPEECH SYNTHESIZER

The Color Computer is one of the best things to happen to computing since the SS-50 bus! At the price, there is no better way to get into personal computing, and we are now offering a speech synthesizer for the CoCo! With most of the same features of our VS-1, the SP-1 can give your CoCo the power of unlimited speech. Unlike software approaches to speech, the SP-1 does not eat your memory, and provides the clearest, easiest to understand speech we have heard!

The SP-1 is a completely assembled and tested unit which plugs directly into the ROM-PACK slot on the side of the computer. Unlike the VS-1, however, no external speaker is required, as the SP-1 routes its audio directly through the computer and into your television monitor!

We also have a version of our VOX-EDITOR to let you build your voice messages quickly and easily, and then save them to cassette when you are done. The speech files can later be read by Basic in your application programs! The comprehensive manual we supply gives you all the information you need to make your CoCo talk!

SP-1 pack and manual $179.95
SP-1 manual only (price applied to SP-1 purchase) . . . . . $ 19.95

TRANSLATOR FOR SP1

Our SC-1/COLOR translator uses the same highly accurate algorithm as our SC-1 for the VS-1, but drives the SP-1 on the color computer. The SC-1/COLOR, however, is even easier to use! When the SC-1/COLOR translator is loaded, it becomes a part of Basic! To translate and speak English text, numeric data (even punctuations if you desire), you simply program PRINT #1 followed by your text. The translator takes it, converts it to a phoneme stream, and drives the SP-1 directly!

SC-1/COLOR cassette and manual (requires SP-1 for operation) $29.95
(manual not available separately)

SSB UTILITIES DISK

Two of the shortcomings of SSB DOS are that, one, with six-character file names and default extensions it's hard to know from a name what is in a file, and, two, with no disk information sector the only way to know what is on a disk is to list the files! Since we couldn't conveniently make the file names longer, we did the next best thing and wrote a family of programs to allow you to 'name' your disks!

Utilities #1 provides a NAME utility to build a disk information sector on each newly-formatted disk. UPDATE lets you change this sector. LOCK lets you write, delete or even LIST protect the files on the disk. LIST is a better listing utility which reads the disk access code from the information sector and will only list LOCK'ed files if you supply the access code. A number of other programs are also included on the disk.

The manual provides information about the way SSB disks are formatted and how the chained sector method work in addition to complete descriptions about each of the programs provided on the disk!

UTILITIES #1 disk and manual $29.95
(manual not available separately)

FANTASTIC PROOFREADER

We are proud to offer Pete Stark's Spell 'n Fix proofreader. We examined a number of very good pieces of spelling correction software. We compared ease of use, speed, correction capability, dictionary maintenance and expansion, disk space overhead, etc..., and found that Pete Stark of StarKits has done a remarkable job on this program. Look at these features and make your own decision....

MULTIPLE DICTIONARIES — Spell 'n Fix allows you to create multiple dictionaries. In this way, if you want, you can put technical and special words in a separate dictionary. Then, when you proofread your documents, select the dictionaries which have the words you need, thus speeding up the job!

EXECUTE DOS COMMANDS — Need to list your disk to decide which dictionary or text file to proofread? Want to delete an old file? View a file? Most of your DOS commands can be executed directly from within Spell 'n Fix.

MULTIPLE CORRECTION OPTIONS — With Spell 'n Fix, you can add new words to your dictionary at any time. Or, if you want, you can create a new dictionary for words not found in the existing dictionaries. Or simply print out the new words. Or even ignore them! CORRECTION OR MARKING - With Spell 'n Fix, you can mark incorrectly-spelled words in your text for later correction, or you can tell Spell 'n Fix to correct them as they are found. The choice is yours!

COMPRESSED DICTIONARY — Spell 'n Fix uses either of two forms of dictionary, and converts either to the other at your command. The first type is a fully expanded dictionary with one word per line, allowing you to use your editor to make corrections, deletions and additions easily. The second type incorporates a slick compression method to drastically reduce dictionary size and increase access speed.

After looking at our options, we decided that this was the program for us, and are proud to be the exclusive agents for StarKits for the SSB DOS68 and DOS69 versions. We can also supply versions for 6800 and 6809 Flex. The OS-9 version will be available soon directly from StarKits. StarKits also has a version for the Color Computer! At the price, we feel that no other program can do what Spell 'n Fix can....

SPELL 'N FIX disk (with 17000 word dictionary) and manual $89.29
(manual not available separately)

SS50 SPEECH SYNTHESIZER

The VS-1 is a completely-assembled, tested unit for the SS-50 I/O bus which gives your computer the power of unlimited speech. Since the VS-1 is programmed phonetically, there is essentially no limitation on what you can make your computer say! The VS-1 programs just like a parallel printer driver, and requires an average of only one phoneme per letter in English for a very low message memory storage overhead. The data rate is typically a low 70 baud.

The VS-1 is supplied with the easiest-to-use, most flexible speech editing program available. This program makes building your speech files or data a real pleasure. We also supply additional software and sample speech files on the disk to give you a 'leg up' on making your computer talk.

The manual provides sections on speech generation theory, operation of the supplied software, complete technical descriptions, a sample speech dictionary and phonetic code charts, fifty pages in all!

If you are a roll-your-own nut, we provide the bare board or manual or disk or synthesizer chip, or combinations (We suggest that only VERY experienced persons attempt such assembly due to the fact that we do not provide assembly instructions beyond parts list parts layout and schematic).

The VS-1 is available for 6800 and 6809 FLEX, 6800 and 6809 SSB DOS, and OS-9 level I and II.

VS-1 complete with manual and disk $194.95
VS-1 manual and disk only (may be adapted for brand-X) . . . . . $ 29.95
VS-1 manual only (may be applied to VS-1 order. Look before you leap!) . . . $ 19.95
VS-1 bare board (build your own...requires experience!) . . . . . $ 19.95
SC-01A synthesizer chip . . . . $ 60.00
VS-1 bare board, chip, disk and manual . . . . $ 99.95

TRANSLATOR FOR VS1

If you have our VS-1 (or even brand-X), and want an easier way to make your computer talk, our SC-1 translator is for you. The SC-1 takes plain English text from a disk file or your keyboard, translates it to a phoneme stream, and sends the resulting output to either the speech synthesizer hardware or to a data file for later use.

The SC-1 translator is one of the most accurate such programs we have seen, making typically only one 'mistake' per sentence, for an average accuracy of about 90%! Due to the type of algorithm we are using, even proper names are generally translated correctly! Where the SC-1 is being used to compile 'canned' messages for later use, we provide an additional program which takes a binary phoneme file created by the SC-1 and converts it to a speech file which may be edited by our VOX-EDITOR (supplied with the VS-1). In this way, you can quickly create your messages and then 'fine-tune' them!

The SC-1 translator is available for 6800 and 6809 FLEX, 6800 and 6809 SSB DOS, and OS-9 level I and II. If you are using brand-X, order the SC-1 with the VOX-EDITOR on the same disk!

SC-1 disk and instructions $39.95
SC-1 disk, instructions with VOX-EDITOR $49.95

TREK-68

Our TREK program is the best 'real-time', 'live-action' game available for 68xx computers using memory-mapped displays. This program does not require graphics capabilities from your display. Otherwise, this is a full arcade-type game. No triviality here, either. The suicide option has only been won by a handful of people world-wide, and can run for hours!

TREK is available for 6800 and 6809 FLEX and 6800 and 6809 SSB DOS, and requires both a memory-mapped display and a source of 60 or 100 Hz interrupts (such as the MP-t timer board).

TREK disk and manual $24.95

COLOR-TREK

Like our other version of TREK, COLOR-TREK provides you seat-of-the-pants action. This program uses no routines from Basic, and so can run on non-extended Basic machines. 16k memory is required. Only one player to our knowledge (other than the author) has won the suicide option of this game, and it took him fifty-seven minutes! No triviality here!

COLOR-TREK cassette and manual $17.95

INTELLIGENT PRINTER INTERFACE

For SS-30 and SS-30C Computers (SWTPc, GIMIX, SSB and Others)

COSTS ABOUT THE SAME AS AN ORDINARY INTERFACE. Our Intelligent Printer Interface offers much more. First, it features an on-board MC6802, 2K bytes firmware and 2K bytes (expandable to 8K) on RAM buffer. We have both RS-232-C Serial and Centronics parallel versions. Both versions fit on the SS-30 (or SS-30C) bus. They work with standard system software. On-board buffering of print data is automatic and allows print spooling. But, there is more: we have features which can be invoked under software control. Control sequences may be intermingled with print lines.

TEXT FORMATTING: set left/right/top/bottom margins • set page length/page size • set horizontal/vertical tabs.

CONTROL: discard print, halt printer, restart printer, halt at top of page, disable buffering, test buffer, test printer

FILTERS: ignore specified characters, translate characters, auto linefeed, download user-written filter program

ChiraTech Scientific Instruments

204 N. Link Lane. Alpha 9
Fort Collins, Colorado 80524
(303) 484-1913

INTELLIGENT PRINTER INTERFACE

VCM-SP Serial Printer version, assembled, tested Owner's Manual $129.95

VCM-PP Parallel Printer version, assembled, tested & Owner's Manual $119.95

Cable assembly (serial or parallel type printer) $19.95

Please add $3.00 S/H charges per order. Colorado residents add 3% tax. MC/VISA accepted.

JCP (Job Control Program) for 6800/6809 FLEX systems (see our ad in January '83 '68' Micro Journal) $49.95

JCP Source + Object $89.95

FLEX™ is a registered trademark of Technical Systems Consultants, Inc.
Centronics™ is a registered trademark of Centronics Data Corp.

COMPARE

our EPROM PROGRAMMER with the field.

All data taken directly from manufacturer's current advertising. Software, interfaces, or personality modules may also be required at additional cost.

• Triple voltage
• Supplied in kit form

	A	B	C	D	E	F
INTERFACE	S30	PAR	PAR	SER	S30	SER	SER
INTELLIGENT	NO	NO	NO	YES	NO	YES	YES
PROGRAMS							
2704•			•				•
2508	•			•	•	•	
2708•			•				•
2758		•	•	•		•	
2516		•	•	•	•	•	
2716		•	•		•		•
2716•			•		•		•
2532	•		•	•	•	•	•
2732	•		•		•	•	•
2732A	•		•		•	•	•
2564	•		•		•	•	•
2764	•		•		•	•	•
2528	•				•		
27128	•						
2816							•
68764			•				
8748						•	
8749						•	
TOTAL	11	3	12	6	11	11	11
PRICE	$125	$45•	$169	$289	$375	$489	$575

EPROM ERASER/Programmer, $125. Personality module for 2508, 2758, 2516, and 2716 included. Specify CPU, disk size, and operating system (TSC's FLEX or SSB's DOS) when ordering. Manual only, $10; refundable with EPROM purchase.

UNITEK • P.O. Box 671 • Emporia, VA 23847

JPC PRODUCTS CO.

POOR MAN'S FLOPPY

HIGH SPEED CASSETTE SYSTEM

Now for the TRS-80 Color Computer

The JPC PRODUCTS High Speed Cassette System, in operation for over 4 years, is now available for all versions of the Radio Shack™ Color Computer.

- TC-8C — Plugs directly into the expansion port of your TRS-80 Color Computer. It is fully compatible with all versions of the Color Computer from the standard 4K to the Extended 32K.
  - FAST — Twice the speed of the Color Computer System.
  - RELIABLE — Less than one error in a million bits.
  - SUPPORTS TWO DRIVES — Software selectable.
  - ALL FILE TYPES — BASIC, machine language, data.
  - MOTOR CONTROL — Two on-board relays.
  - EPROM OPERATING SYSTEM
  - SPARE EPROM SOCKET — 2716 or 2732 compatible.
  - OPTIONAL JBUG MONITOR — EPROM or Cassette
    - 6809 Assembler
    - 6809 Dis-assembler
    - Memory modify and list
    - Break point traps
  - ASSEMBLED and TESTED

TC-8C ..... $129.95 JBUG (EPROM) $34.95
JBUG (Cassette) $29.95

TERMS:
Cash, Master Card or Visa
Shipping & Handling $3.50 (US)
$5.50 (Canada) $15.00 (Foreign) Technical inquiries: Phone 5:00 - 6:00 PM MST

JPC PRODUCTS CO.
Phone (505) 294-4623
12021 Paisano Ct. NE
Albuquerque, NM 87112

'68' MICRO JOURNAL

★ The only ALL 6800 Computer Magazine.

★ More 6800 material than all the others combined: **MAGAZINE COMPARISON**

(2 years)

Monthly Averages

6800 Articles

KB	BYTE	CC	DOBB'S	TOTAL PAGES
7.8	6.4	2.7	2.2	19.1 ea. mo.

Average cost for all four each month: $6.53
(Based on advertised 1-year subscription price)

'68' cost per month: $2.04

That's Right! Much, Much More
for About
1/3 the Cost!

OK, PLEASE ENTER MY SUBSCRIPTION

Bill My: Master Charge ☐ — VISA ☐

Card # ______________ Exp. Date ______________

For ☐ 1-Year ☐ 2 Years ☐ 3 Years

Enclosed: $______________

Name ______________________________

Stree

City ______________ State ________ Zip ________

My Computer Is: ______________________________

68 Micro Journal
5900 Cassandra Smith Rd.
Hixson, TN 37343

SUBSCRIPTION RATES

USA
1 Year $24.50, 2 Year $42.50, 3 Year $64.50

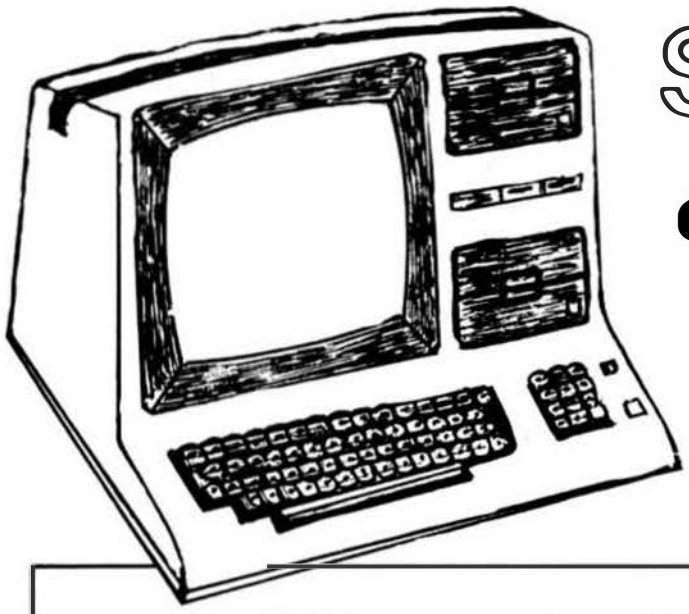
*FOREIGN SURFACE Add $12.00 per Year to USA Price

*FOREIGN AIRMAIL Add $36.00 per Year to USA Price

**CANADA & MEXICO Add $5.50 per Year to USA Price

Cash (USA) or drawn on a USA Bank!!!

Universal Data Research, Inc. Introduces



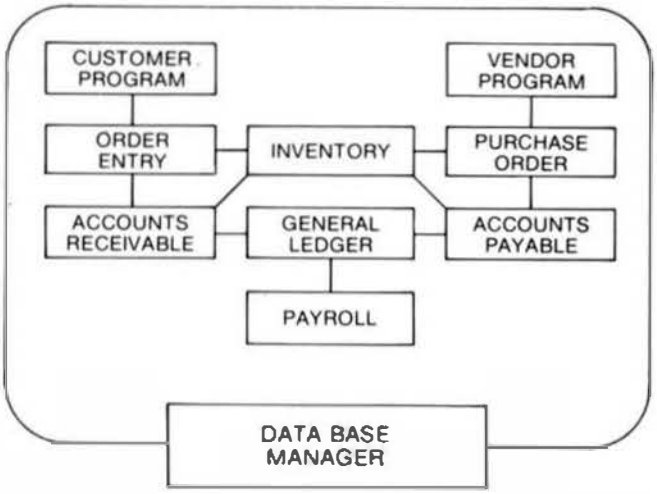
SOFTWARE
for the COLOR COMPUTER and the TRS 80 Model III

- Data Base Manager _____ $150
- Church Contributions Package _ $150
- Single Entry Ledger _____ $ 95

FLEX* and UniFLEX* SOFTWARE for the 68XX Operating System . . .

INTEGRATED BUSINESS PROGRAMS
for large and small companies

	FLEX	UniFLEX
Accounts Receivable	$395	$495
Accounts Payable	$395	$495
General Ledger	$395	$495
Inventory 2	$395	$495
Payroll	$395	$495
Data Base Manager	$450	$550



STAND-ALONE SOFTWARE

- Single Entry Ledger
- Check Balancer
- Costing and Estimating
- Church Contribution Pkg.
- Church School Billing
- Fund Raising
- Custom Programming
- M.F.I. (Extensive Mfg./P.O.)
- Bulk Mailings
- Maintenance Schedule
- Vendor Program
- Purchase Order
- Accounts Payable
- Accounts Receivable
- Customer Program
- Order Entry

UNIVERSAL DATA RESEARCH INC.

*FLEX & UniFLEX are Trademarks of Technical Systems

2457 Wehrle Drive, D-1, Buffalo, New York 14221
Phone (716) 631-3011

master charge VISA

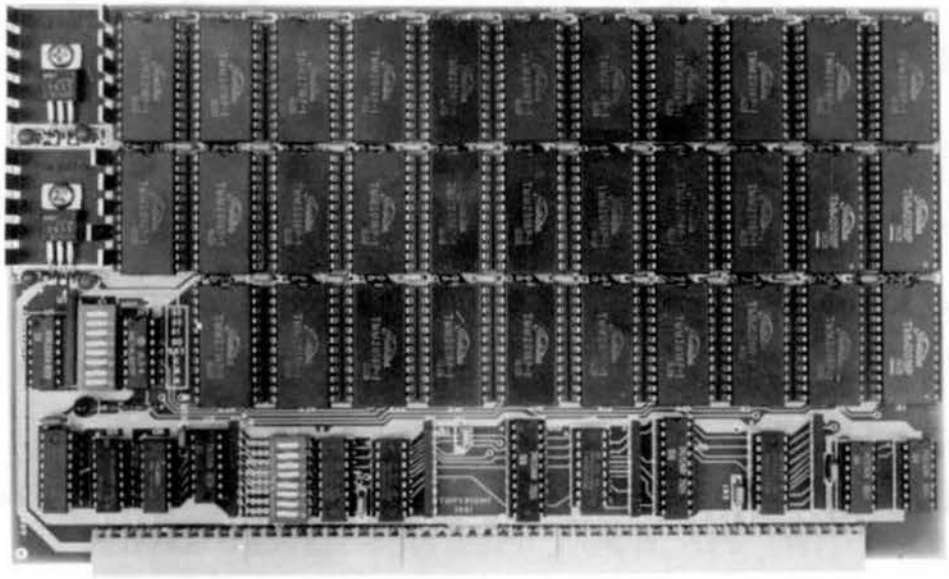
64K SS-50 STATIC RAM

NEW!

$199.00 (48K KIT)

NEW!

LOW POWER!



RAM OR EPROM!

BLANK PC BOARD WITH DOCUMENTATION $52

SUPPORT ICs + CAPS -	$18.00
FULL SOCKET SET -	$15.00

ASSEMBLED AND TESTED **ADD $40**

FEATURES:

- ★ Uses new 2K x 8 (TMM 2016 or HM 6116) RAMs.
- ★ Fully supports Extended Addressing.
- ★ 64K draws only approximately 500 MA.
- ★ 200 NS RAMs are standard. (TOSHIBA makes TMM 2016s as fast as 100 NS. FOR YOUR HIGH SPEED APPLICATIONS.)
- ★ Board is configured as 3-16K blocks and 8-2K blocks (within any 64K block) for maximum flexibility.
- ★ 2716 EPROMs may be installed anywhere on Board.
- ★ Top 16K may be disabled in 2K blocks to avoid any I/O conflicts.
- ★ One Board supports both RAM and EPROM.
- ★ RAM supports 2MHZ operation at no extra charge!
- ★ Board may be partially populated in 16K increments.

56K	$249
64K	$299

16K STATIC RAMS?

The new 2K x 8, 24 PIN, static RAMs are the next generation of high density, high speed, low power, RAMs. Pioneered by such companies as HITACHI and TOSHIBA, and soon to be second sourced by most major U.S. manufacturers, these ultra low power parts, feature 2716 compatible pin out. Thus fully interchangeable ROM/RAM boards are at last a reality, and you get BLINDING speed and LOW power thrown in for virtually nothing.

Digital Research Computers
(OF TEXAS)
P.O. BOX 401565 • GARLAND, TEXAS 75040 • (214) 271-3538

TERMS: Add $2.00 postage. We pay balance. Order under $15 add 75¢ handling. No C.O.D. We accept Visa and MasterCharge. Tex. Res. add 5% Tax. Foreign orders (except Canada) add 20% P & H. Orders over $50. add 85¢ for insurance.

ARCADE 50

POWERFUL COLOR GRAPHICS
Uses the new TMS9918A Video Display Processor. High resolution 256 x 192 pixel display with 15 colors 16k Bytes of onboard RAM does not reduce user memory. 32 graphic images can be individually moved with simple X-Y commands for smooth animation
External Video input allows subtitling
NTSC composite video output
SOUND EFFECTS AND MUSIC
Three AY3-8910 Programmable Sound Generators
Nine simultaneous voices
Three independent noise sources
Onboard stereo amplifier drives two 8 ohm speakers
ADDITIONAL I/O CAPABILITIES
Eight analog inputs with 8 bit resolution
Supports four joysticks with pushbutton switches
Eight bit parallel I/O port
Entire unit maps into 256 bytes of memory
DOCUMENTATION AND SOFTWARE
Programming manuals for Video and Sound Processors
Subroutine library and Super Demo Maze Game
Example programs in BASIC, FBASIC and ASSEMBLY
User library and sales support

ARCADE 50 assembled and tested	$325.00
Video and Audio connector set	15.00
4 Joystick connector set	15.00
2 Radio Shack Joysticks	24.00
UHF channel 33 modulator	32.00
Gold Molex connectors	12.00
A/BASIC for 6800	110.00
FBASIC for 6809	110.00
FBASIC (with ARCADE 50)	75.00
FBASIC (manual only)	10.00
ARCADE 80 (TRS Model I)	395.00
ARCADE 100 (S-100 BUSS)	375.00
ARCADE 50 RGB	375.00
LABVIDEO (Motorola EXORbus)	375.00
LABVIDEO RGB	375.00
NEW MV09 6809 Processor Board	225.00
•Comes assembled with PIA and ACIA	
•12 Sockets for 2716, 2732 or RAM	
•Supports DMA disk I/O	
•Ideal for 6809 upgrade or process control	
AMDEK COLOR I Monitor	425.00
AMDEK COLOR II Monitor	799.00
AMDEK COLOR III Monitor	499.00
256K Dynamic Memory Board (assembled)	795.00
256K Dynamic Memory Board (assembled w/64K)	395.00
64K Dynamic Memory Board (assembled)	295.00

Specify 5" or 8" soft sector disk for TSC's FLEX or MICROWARE'S OS/9 system
TERMS: CASH, VISA, MC, C.O.D.

FBASIC

TERMINUS DESIGN INC., in conjunction with Microware Systems Corporation, is proud to announce FBASIC-an enhancement of Microware's 6800 A/BASIC. Their fast compiled BASIC has been adapted for 6809 users with added video and sound features for ARCADE 50 users. FBASIC is a true compiler that produces optimized machine language modules which are ROMable and require no Run-Time package. FBASIC requires less memory overhead and runs hundreds of times faster than BASIC interpreters. It supports standard BASIC instruction including String functions, Disk I/O and fast integer arithmetic with multiple-precision capability. Graphics verbs and functions fully support the Arcade 50. Arcade statements include:

INIT	MODE	BLANK	BACKDROP
SIZE	MAG	VREG	DELAY
MOVE	DRAW	FCOLOR	JSWITCH
REMOVE	RDRAW	BCOLOR	SWITCH
PSG	TONE	ENVL	VOLUME
ADC	SPRITE	SPNAME	ENDEF
SPCOLOR	RSPRITE	SPDEF	PATDEF
VPEEK	VPOKE	VPRINT	

TERMINUS DESIGN INC
16 SCARBROUGH ROAD
ELLENWOOD, GA 30049
(404) 474-4866

WESTCHESTER Applied Business Systems
Post Office Box 187, Briarcliff Manor, New York 10510

~~~ 6809 SOFTWARE ~~~

DMS2/VM DATA MANAGEMENT SYSTEM
A complete Data Management System which permits 1+ Megabyte files, precision BCD arithmetic, Multi-key access, selection and sorting. DMS2/VM employs a virtual memory access method under which programs "think" that entire files are in memory and directly accessible. The system supports alphanumeric, numeric, decimal, integer, coded, hex and scalar data field types. Up to 2e fields and 12 levels per file may be defined by the user. A simple high-level command language allows a variety of data manipulation including reformatting, calculations, inquiry, key-merge, summation, displays and customized reports.
DMS2/VM DATA MANAGEMENT SYSTEM (5" or 8" disk) .....$ 99.95
DMS2-UG DMS2/VM User Guide Only .....$ 14.95
DMS2-US DMS2/VM System after user guide purchase (not incl).....$ 89.95
DMS2-UV DMS2/VM New Version Upgrade (existing users only) .....$ 25.00

ACC2/VM GENERAL ACCOUNTING SYSTEM
All essential accounting and bookkeeping functions including journal, ledger, income statement and balance sheet. The user defines accounts, products and transactions to the system and thus tailors it to his own retail, wholesale or service environment. The system operates under DMS2/VM which permits custom reports of product movement or account status to be generated. Accounts receivable and payable are integral to the system as is point-of-sale register capability.
ACC2/VM GENERAL ACCOUNTING SYSTEM (5" or 8" disk) $249.95
ACC2-UG ACC2/VM User Guide Only .....$ 14.95
ACC2-US ACC2/VM System after user guide purchase (not incl).....$239.95
ACC2-UV ACC2/VM New Version Upgrade (existing users only) .....$ 25.00

GENERAL PURPOSE UTILITIES
DUMP - Output any section of memory in hex and ASCII. DDSK - Output any disk sector in dump format. DMAP - Output track & sector chain of any file. PMAP - Output load map of command files. LISTD - Output all directory info on files. LISTDS - Output selected directory info in three columns. LISTF - List file with disk id & date heading. KILL - delete files without "are you sure" prompt. FIND - Output all file records containing a given string. MCOPY - Copy files between disks using one drive. ZAP - Interactive sector display/repair. LISTM - list files in multi columns. CUT - Assembler programmer aid. RESIZE - Reformat text line sizes. TYPESET - Spread text between margins. MSG - Output text/hex message. LOGO - Output string as large 5 x 8 title. WAIT - Delay execution. SKIP - Output line feeds. MOVE - Relocate block of memory. Utilities output to CRT, printer or disk where applicable. Each is shipped with a one page instruction sheet suitable for filing alphabetically or by function.
UTIL-1 SET OF 20 GENERAL UTILITIES (5" or 8" disk) .....$ 59.95

All software is written in modular assembler and runs under FLEX O/S, DMS2/VM and ACC2/VM new version upgrades on "as available" basis. Terms: Check, Money Order, Mastercharge or Visa. Shipment first class. Add P&H $2.50 ($7.50 Foreign). N.Y. Res add sales tax. Specify 5 or 8". Sales: 212-899-0400 (Ask for Sandy) / Consultation: 914-914-3532 (Eves). Dealers & OEM's - Quantity discounts available - Write for details.

WINCHESTER FOR MOTOROLA EXORCISOR/MDOS

☐ 10 MB Winchester hard disk runs MDOS on Motorola Exorcisor System ☐ No modification to MDOS required ☐ MDOS based software stays alive ☐ All user software operates without modification ☐ Optional SA-801R flexible diskette drive system ☐ Optional 10 MB removable cartridge.

For information call (619) 566-3911
Computer System Associates
7562 Trade Street, San Diego, CA 92121


~~~

AAA Chicago Computer Center

ELEKTRA CABINET Made of heavyweight 0.090" thick aluminum. Interior is 18-1/2" wide by 21-7/8" deep by 6-3/4" high. Heavy duty A.C. line cord. A.C. fuse holder. EMI filter. Fan with filter. Back panel has 10 cutouts for 'D' type data connectors. Front panel has key on/off power switch, 2 illuminated push button switches (Reset and NMI/Abort), and two cutouts for 5-1/4" disk drives.

POWER SUPPLY Highest quality linear power supply CONSERVATIVELY rated at 15a @ 8v, e @ 16v, 3a @ -16v. 3 primary inputs for light, rated, and heavy loading.

DISK REGULATOR BOARD WITH CABLES Standard version for 2 floppydrives. Heavy duty version for 1 Winchester drive and 1 floppy drive.

ELEKTRA UNIVERSAL MOTHERBOARD Heavyweight 0.125" thick. 18" long by 8" wide. 11 memory (50 pin) slots. 4 or 8 slots may be cut off for shortening to 14" or 10" lengths respectively. 8 I/O (30 pin) slots. Complete address decoding of I/O slots. Choice of 4, 8, or 16 addresses per I/O slot. Base address for I/O slots can be placed at 32, 64, or 128 byte increments respectively. 1" spacing between all memory and I/O slots. Extended addressing capability for both memory and I/O ports for meeting SS-50C bus specifications. On board baud rate generator with low and high ranges providing jumper selectable rates of 75 through 38,400 for each of the five baud rate lines. Slow device circuitry permitting 1MHz 30 pin disk controllers to run with 2 MHz 50 pin CPU boards.

ELECTRA CPU 8/9 Use either the 6802 or 6808 (to run 6800 software) or 6809. Has provision for up to 3 2716 Eproms, 1K scratchpad, MC6840 triple timer, and an optional baud rate generator providing baud rates from 110 through 38,400 baud in two user selectable ranges. The board supports DMA by either HALT or BUSREQ when a 6809 CPU is used. The board does not support a DAT and therefore does not support extended addressing. The board will run any of the MIKBUG™ compatible monitors in the 6802/6808 mode and SBUG-E, HUMBUG, and MICROBUG in the 6809 mode. The ELEKTRA CPU 8/9 will run any of the popular disk controller boards with the appropriate software. OS-9™ Level 1 is available when used with the SSB DCB-4A controller or Gimix compatible single density controllers.

ELEKTRA OPS DUAL PORT SERIAL CARD Fits the standard 30 pin SS-50 bus I/O slot. Can be configured for 4 addresses per port with the B port 2 addresses higher than the A port or for 16 addresses per port with the B port 4 addresses higher than the A port. Each port is terminated at two 16 pin dip sockets, one socket configured for modem and the other for terminal or printer. RTS, CTS, DTR, DCD, and DTR are appropriately implemented. Each port has independent selection of baud rate. Each port allows the interrupt request to be independently jumpered to the IRQ or FIRQ/NMI bus line.

ELEKTRA DPP DUAL PORT PARALLEL CARD Fits the standard 30 pin SS-50 bus I/O slot. Can be used in either the 4 or 16 addresses per I/O slot configuration occupying the first four addresses of the I/O slot. The direction of the TTL buffers can be controlled by either on board jumper connectors or by a signal from the peripherals. The interrupt request lines for each port may be individually jumpered to either the IRQ or FIRQ/NMI bus line.

ELEKTRA CHASSIS Includes cabinet, 110v power supply, power supply cables, standard disk regulator board with power cables, assembled motherboard with gold square pin connectors	$825.00*
ELEKTRA CABINET	260.00
ELEKTRA 110v POWER SUPPLY	175.00*
ELEKTRA CABINET, 110v POWER SUPPLY, AND POWER SUPPLY CABLES	410.00*
ELEKTRA CABINET, 110v PS, PS CABLES, STD. DISK REGULATOR AND CABLES	460.00*
*Add $30.00 for 220v	
STANDARD DISK REGULATOR BOARD AND CABLES	50.00
HEAVY DUTY DISK REGULATOR BOARD AND CABLES	75.00
FILLER PLATE FOR 5-1/4" DRIVE OPENING	10.00
FAN FILTER	10.00
ELEKTRA D-5 Dual drive cabinet for 5-1/4" drives with power supply, line cord, fuse, power switch, and power cable to drives	125.00
ELEKTRA HD-5 (Heavy duty version of D-5 package above)	150.00
ELEKTRA SHD-5 (Super heavy duty. Powers 1 Winchester and 1 floppy)	175.00
5" ribbon cable for dual 5 1/4" disk drives	0.00
ELEKTRA D-8 Dual drive cabinet, power supply, ps cable for 8" drives	350.00
Cabinet for dual 8" drives only	250.00
Power supply for dual 8" drives only	120.00
PS cables only (Specify brand and type of 8" drives)	30.00
6' ribbon cable for dual 8" disk drives	45.00
GOLD 10 PIN CONNECTORS (Specify male w/square pins or female)	1.50
TIN 10 PIN CONNECTORS (Specify male w/square pins or female)	0.50
TITANIUM-TIN 10 PIN FEMALE CONNECTORS (While supply lasts)	0.50

	BARE BOARD	KIT	ASSEMBLED
ELEKTRA UNIVERSAL MOTHERBOARD (Gold)	80.00	320.00	380.00
ELEKTRA UNIVERSAL MOTHERBOARD (Tin)		240.00	300.00
ELEKTRA CPU 8/9 (Add $20.00 for MICROBUG)	50.00	225.00	275.00
ELEKTRA CPU 8/9 baud rate option		25.00	25.00
ELEKTRA DPP DUAL PORT PARALLEL BOARD**	20.00	60.00	80.00
ELEKTRA DPS DUAL PORT SERIAL BOARD**	20.00	60.00	80.00
**CABLE FOR DPP OR DPS (2 needed, specify board)			20.00

WARNING AAA Chicago Computer Center does not provide repair or diagnostic service for customer assembled kits. AAA Chicago Computer Center does warranty and maintain service for our assembled boards. The customer should carefully take into consideration the small differential separating our kit and assembled prices when making his choice of purchase.

We have introduced our line of computer equipment with the purpose of offering the highest quality of components possible at affordable prices. These products are intended for OEM applications where it is the responsibility of the purchaser to integrate these components with suitable memory, disk controllers, drives, and software along with I/O terminals to form working computer systems.

AAA Chicago Computer Center

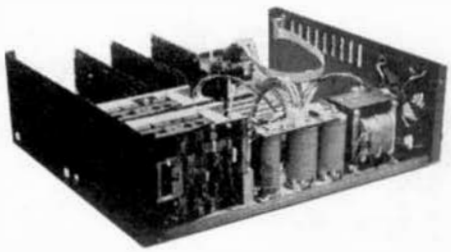
120 CHESTNUT LANE • WHEELING, IL 60090
(312) 459-0450

Technical consultation available 4 PM to 6 PM most weekdays. Closed evenings and weekends.

TERMS Minimum order $20.00. Shipping and handling estimates within the Continental U.S., add 3% (MINIMUM $2.50). Illinois residents add 6% sales tax. We will refund your overestimated shipping and handling charges. Foreign shipping and handling, add 10% (MINIMUM $10.00). Foreign orders must be prepaid in U.S. dollars. Heavy foreign items will be shipped air freight collect. Please phone between 4 PM and 6 PM weekdays if questions arise regarding shipping fees. Master Charge, Visa, and American Express honored.

Our apology. We are not staffed to answer technical inquiries through the mail. Please phone for technical help during the hours indicated above. The too frequent changing of our inventory and prices makes it uneconomical to publish a catalog. Our ads are intended to serve that purpose. Prices and inventory are subject to change without advance notice.

ELEKTRA
COMPUTER PRODUCTS



SMOOTH™ Software

(All of our software is copyrighted and all rights are reserved. Source is either supplied or optionally available at extra cost so that the purchaser can modify our programs for his own use. Licensing, however, is required for commercial resale.)

SUPER MODEM PROGRAM

Hardware independent (No interrupts required). Assumes modem is connected to an MC68 0 (serial interface) and the control port is connected to either an MC6850 or MC6820 when used with the Video GMXBUG.

Transmit manuaay to distant computer.

Transmit disk files (text) of any length to distant computer.

Receive and save disk files (text) of any length on local disk system. If sending computer does not support an X-on/X-off protocol, then the received files are limited in size by the computer memory.

Tested to transmit and receive text at speeds up to 9600 baud. (CRT terminal must be capable of operating at a baud rate higher than the one the modem is operated at.)

Halt duplex option in case distant computer doesn't echo.

Echo option so user can simulate a time sharing system. (Super Modem Program doesn't support auto-answer but the source is provided for those individuals who wish to adapt our program to their special needs.)

Replaces CR with CR/LF (user option) for those using time sharing systems that don't transmit LF's.

Slow disk file transmit based on character verification (plus user installed timing loops if necessary) for use on time sharing systems to which disk files cannot be sent at speeds suggested by the baud rate.

Please specify 6800 or 6809, SSB or FLEX™, 5" or 8"
Manual and disk with both source and object code $75.00

STANDARD MODEM PROGRAM

Same as Super Modem Program above but without ECHO option, CR/LF for CR option, slow disk file transmit option, nor X-on/X-off option. Reception of disk files is limited to those small enough to completely fit within the receiving buffer.

Please specify 6800 or 6809, SSB or FLEX™, 5" or 8"
Manual with instructions, source listing, and flow chart; disk with both source and object code $45.00
Manual with instructions, source listing, and flow chart 25.00

MODEMS (By U.S. Robotics) Auto-Dial is Hayes compatible	MANUAL ANSWER	AUTO ANSWER	AUTO-ANS AUTO-DIAL
300/1200 Baud, direct connect	N/A	499.00	549.00
1200 Baud (120 cps), direct connect	399.00	449.00	N/A
300 Baud (30 cps), direct connect	159.00	199.00	N/A
300 Baud (30 cps), acoustic	129.00	N/A	N/A

SMOOTH™ Software

ALL IN ONE
Editor - Text Processor - Mailing Labels
Mailing Lists - Use any CRT terminal and printer

Supports Editing commands such as bottom, change, delete, find, insert (single line), input (multiple lines), list, next, overlay (with cursor editing, character deletion and insertion), overstrike (for selected darker text), print, restart, set, top, undefine, up, and verify.

Supports Text Processing commands such as block copy, block move, centering, margin justification (widen and narrow), paging, and tabbing.

Mailing Lists and Labels. Use the same mailing list disk file (with protected areas) for both mailing labels and repeat letters. Repeat letters are personally adressed to each person or selected persons on the mailing list.

Most Powerful File Handler found in any editor. Append one file to the end of another or insert (merge) one file into another as designated by the fine pointer. Print specified lines to your printer or to a disk file. Edit files larger than the text buffer. Does not produce output files when not desired. Delete disk files from the editor.

Printer commands. Control characters can be sent to the printer for format control either directly from the control terminal or by imbedding them in the text. The set command contains interface initialization and character output routines to support the SWTPC MP-C interface as well as the standard serial and parallel interfaces. Jumps are also provided to user supplied printer routines. Users elects the port address (0 thru 7, A or B) thereby eliminating the need for the user to install printer software routines. Editor can be initialized for either 4 or 16 addresses per port.

Editor allows exiting to either the monitor or DOS and then reenter (Warm Start) without destroying previously prepared text in the buffer. The Restart command erases contents in the buffer without the user having to reload the Editor.

The Editor allows the user to toggle between full duplex (no echo) and half duplex (echo) as needed. It responds to commands in both upper and lower case and can be used to create assembler source code and Basic programs as well as text.

Specify 6800 or 6809, SSB or FLEX™, 5" or 8"	75 00
Printed source listing is available for an additional	35 00
All-In-One, Write'n Spell, and Spell'n Fix package	195 00

Software by Technical Systems Consultants, Inc.	UniFLEX™ w/1 yr. mnt.	FLEX™
DOS (includes Editor and Assembler)	550 00	150 00
Editor		50 00
Assembler		50 00
68000 Cross Assembler on 6809	300 00	250 00
6809 Cross Assembler on 6800		100 00
Text Processor	150 00	75 00
Extended Basic	200 00	100 00
Basic Precompiler (specify standard or extended)	150 00	50 00
Pascal (Flex™)	300 00	200 00
Sort/Merge Package	150 00	75 00
6809 Flex™ Utilities	150 00	75 00
6800 Flex™ Utilities		100 00
Debug Package	75 00	75 00
Diagnostic Package	75 00	75 00
6809 Relocating Assembler & Linking Loader	175 00	150 00
Fortran (Requires Relocating Assembler & Linking Loader)	350 00	275 00
Fortran (With Relocating Assembler & Linking Loader)	450 00	375 00
Cobol	750 00	500 00

Software by Microware Systems Corp.	Run-Time Package	Update	Source	Manual	Object
OS-9™ Level One Operating System		75 00	400 00	40 00	200 00
OS-9™ Level Two Operating System		75 00	N/A	40 00	500 00
BASIC09™	100 00	75 00	N/A	25 00	200 00
OS-9™ Macro Text Editor			300 00	15 00	125 00
OS-9™ Interactive Assembler			300 00	10 00	125 00
OS-9™ Interactive Debugger (Disk version)			100 00	10 00	50 00
CIS Cobol Compiler	400 00	50 00	N/A	80 00	900 00
Pascal Compiler	100 00	100 00	N/A	40 00	400 00
Microware yearly support service ($200 00 for OS-9 Level 2)					75 00

SWTPC	KIT	ASSEMBLED
6809 SWTPC FLEX™ Disk and manual (Disk only 15 00)		35 00
DC-4 Disk Controller (SS/DS, SD/DD, 5-1/4")	N/A	230 00
SBUG-E (2716 compatible)		25 00
MP-S2 Dual port serial or MP-L2 dual port parallel	N/A	120 00
MP-R Single voltage 2716 prom programmer	N/A	114 50
MP-N Calculator board	54 95	92 00
MP-T Interrupt timer	N/A	92 00
532 Universal Static Memory Board	N/A	124 50
MP-09 6809 CPU board (2MHz)	N/A	285 00

Special Software

MICROBUG 12K, 6809 Baby HUMBUG by Peter Stark	30 00
4K 6809 HUMBUG	75 00
4K 6800 HUMBUG (RAM needed at $A000 and $D000)	65 00
2K 6800 HUMBUG (With cassette LOAD and DUMP)	40 00
2K 6800 HUMBUG (Extra commands instead of cassette software)	40 00
Other HUMBUG versions including video versions are available.	
Spell'n Fix by Peter Stark	69 29
Write'n Spell by Peter Stark	75 11
All-In-One, Spell'n Fix, and Write'n Spell package	195 00
Dynamite Disassembler	60 00
Dynamite + (Update. Send original diskette plus $40 00)	100 00
SUPER SLEUTH Disassembler System ($101 00 for OS-9 version)	99 00

SMOOTH and ELEKTRA are trademarks of AAA Chicago Computer Center
FLEX and UniFLEX are trademarks of Technical Systems Consultants, Inc.
OS-9 and BASIC09 are trademarks of Microware Systems Corp.
GIMIX and GHOST are registered trademarks of GIMIX Inc.
CP/M is a registered trademark of Digital Research Inc.
Z80A is a registered trademark of Zilog, Inc.

Dealer for GIMIX, SSB, SWTPC, Microware Systems Corp and Technical Systems Consultants, Inc.

Prices and inventory are subject to change without advance notice. This ad is our catalog.

See our ad on the previous page to your left for ordering instructions.

SPECIALS

- New Televideo Z80A, 4MHz, 64K computer system with 950 compatible built-in terminal, dual 5-1/4" floppy drives (1 Mbyte capacity total), CP/M — 3295 00
- Same but with one floppy and one 10Mbyte Winchester — 5495 00
- Six user versions available — Call
- While Supply lasts, titanium-tin 10 pin female connectors — 0 50
- SWTPC MP-09B, socketed, with baud rate generator (only 3 left) — 250 00
- SWTPC MP-A2, socketed, all with crystal clock 1 MHz (only 4 left) — 115 00
- SSB BFD Controllers (only 5 left) — 299 00
- GIMIX 32K Memory Boards with 32K RAM (only 4 left) — 199 00
- GIMIX e78 Video Board (only 1 at this price) — 399 00
- Remex 8" DS/DD Disk Drives (only 4 left) — 399 00
- MICROBUG (2K, 6809, Baby HUMBUG by Peter Stark) — 30 00
- Cat 300 Baud Acoustic Modem (only 1 left) — 99 00
- GIMIX 6800 CPU w/baud rate and timer, used (only 1 left) — 199 00

DISK DRIVES 30 day guarantee SD/DD Capability	1 head MPI	flippy Siemens	2 heads MPI	2 heads QUME
5-1/4", 40 tracks	250 00	260 00	325 00	350 00
5-1/4", 80 tracks	325 00	335 00	425 00	450 00
Service Manual (Specify 40 or 80 track)			20 00	35 00
8", 77 tracks SD/DD Qume (Remex $399 00)				475 00
Service Manual (Qume DT-8)				0 00

SPECIAL BOARDS	
Microtime II Calendar and Clock Board	60 00
Data Mart 16K EPROM bareboard (2708 chips)	30 00
Data Systems "68" Extender Board (Specify 30 Pin or 50 pin)	25 00

TERMINALS	
Hazeltine 1420	399 00
Hazeltine Espirit I (EsPirit II $549 00)	499 00
Adds Viewpoint Green Screen	525 00
Televideo 925 Green Screen	749 00
Televideo 950 Green Screen (Std. Keys $895 00)	949 00
Televideo 950 Black Screen	795 00

Printers	
Okidata ML 82A (120 cps, 9x9, bidirectional, serial and parallel)	475 00
Tractor for ML 82A	80 00
Okidata ML 83A (120 cps w/in tractor)	775 00
2K Byte buffer (High speed RS-232)	125 00
Dot addressable graphics	50 00
Okidata ML 84, 200 cps, 2K, GraPhics, Parallel (Serial $1295 00)	1 195 00
TI 810 w/lower case and full vertical forms control (limited quantity)	1400 00
Florida Data (600 cPs)	3495 00
NEC 3510 Std.	1495 00
NEC 7710 Std.	2445 00
Spare ribbon cartridge for Epson MX-80(FT) (While supply lasts)	7 99

Optimal Technology, Inc.	
EP-2A-79 Eprom Programmer (Personality Modules extra for above programmer)	169 00
Optimal Technology, Inc. 30 pin parallel I/O board for EP-2A-79	37 00
Software package for EP-2A-79 (Specify 6800 or 6809)	30 00
Windrush Eprom Programmer for SS-30 (Add $20 00 for Exorcisor)	375 00

Smoke Signal Broadcasting	
DCB-4A Double Density Controller Board for 5" and 8" with DOS	549 00
SSB DOS (Specify 6800 or 6809, BFD or DCB-4A, 5" or 8")	75 00
SE92/SA92-5 (6809 Edit/Assm for DOS)	69 95
SSB Monitor (Specify 6800/6809, $8000/$E008/$F7E8)	75 00
SSB version of FLEX™ (without Editor and Assembler)	150 00
LMB-1A Motherboard	399 00
SCB-69 6809 CPU Board	399 00
PAR-1 Dual Port Parallel Board	89 00
SER-2 Dual Port Serial Board with 2 Cables	129 00
Chief 80 64K Computer System	2195 00
Chief 9524 64K Computer System with DS-DT-DD 5" FD	4325 00
Static Memory Boards M-16-X 195 00 M-24-X 295 00 M-32-X 395 00	
Dynamic Memory Boards M-128-X 995 00 M-256-X 1295 00 M-512-X 1895 00	

GIMIX	
6800 CPU Board(Timers $64 03, Baud rate $ 00, 2 MHz $ 15 00)	224 03
2 MHz 6809 Plus CPU, time of day clock, battery backup, 1K NMOS RAM	578 05
GIMIX Dynamic Address Translator (SWTPC Compatible DAT $15 00)	35 00
Filler plate for 5-1/4" drive opening	14 92
Baud rate generator board	88 93
Missing cycle detect card	38 23
50 Pin Prototyping board (30 Pin Prototyping board $38 33)	56 66
5" Single Density Controller (Without 1771 chip $158 38)	198 48
5" and 8" Single Density Controller	226 58
5" Double Density Controller	298 28
DMA 5" AND 8" Double Density Controller	5 68
GIMIX version of FLEX™ (without Editor and Assembler)	90 00
Double disk regulator card	68 22
Ribbon Cable: Two 5 1/4" inboard drives $34 96. Two 8" outboard drives	44 26
8" disk ribbon cable and back panel connector set	29 25
8" disk drive cabinet with power supply	848 18
19MB Winchester and controller update (38MB $6688 91)	4288 90

Memory	CMOS WITH BAT. BACKUP	NMOS NO BAT. BACKUP
64K Static RAM Board with 24K of RAM installed	N/A	348 27
64K Static RAM Board with 32K of RAM installed	518 36	398 37
64K Static RAM Board with 48K of RAM installed	N/A	518 47
64K Static RAM Board with 56K of RAM installed	728 56	578 57
64K Static RAM Board with 64K of RAM installed	798 64	638 67
16 Socket EPROM/ROM/RAM Board		238 32
8K Promboard (2708)		98 34

I/O Boards	1 port	2 port	8 port
Serial interface	88 41	128 43	318 46
Parallel interface	88 42	128 44	198 45
Cable sets for above boards (specify board)		each	24 95

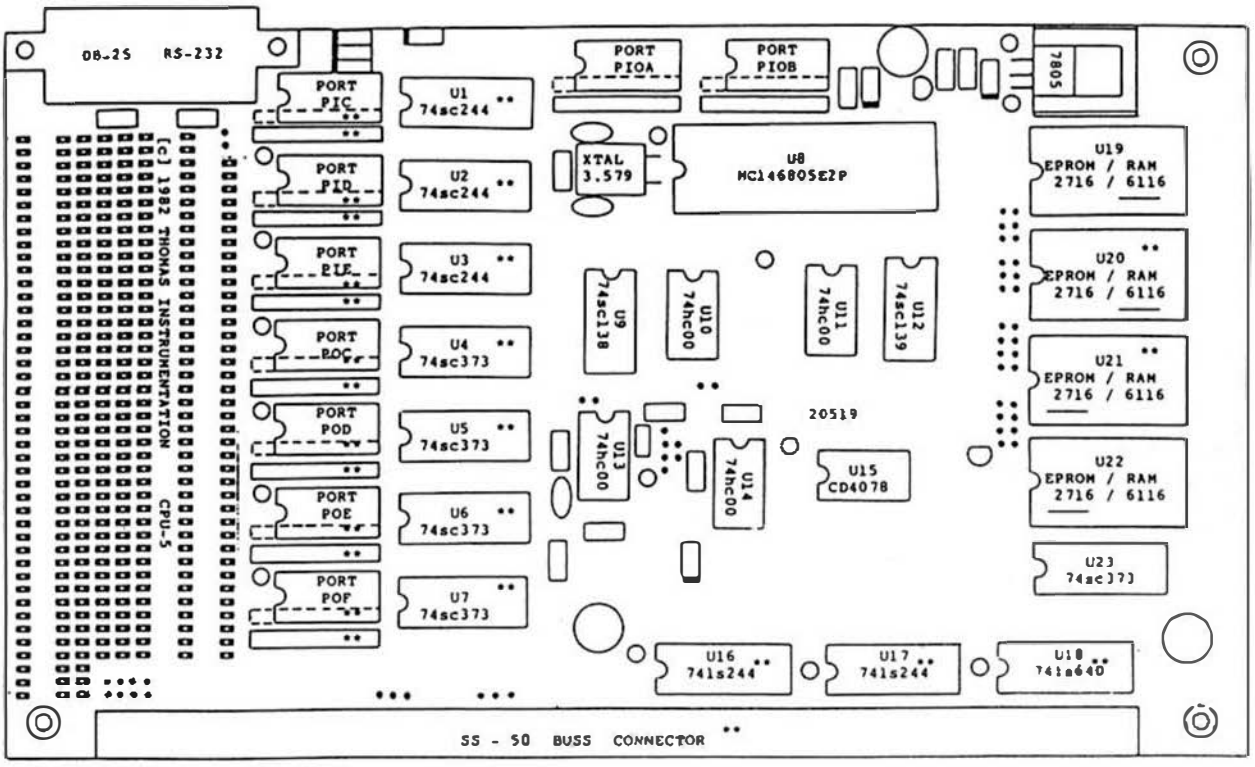
2MHz 6809 PLUS Computer System w/o Disk Cont.	w/#58 Cont.	w/#88 Cont.
With 56K of memory 2498 29	2988 59*	3248 49*
With 128K of memory		3798 39
With 128K and 19MB Winchester		8998 09
With 128K, 38MB Winch., dual 8"		17498 99

*Includes GMXBUG/FLEX/OS-9 software selectable

*With CMOS RAM and Battery Backup	add 150 00
Mainframe (Chassis, PS, Switches, Fan, Motherboard, Baud Rate Gen.)	1198 19

AAA Chicago Computer Center

*** NEW 6805 ALL CMOS CARD ***



CPU-5-B	Blank card with documentation	$49.00	
CPU-5-P	Partial card (parts marked ** not incl.)	$229.00	Includes EPROM Monitor
CPU-5-F	Full card with all parts shown (fully assembled)	$349.00	and 115 vac Power Pak.
CROSS-ASSEMBLERS	for FLEX ([c] TSC) 6809 system	$150.00	

SOFTWARE

VDISK Treat extended memory as a super fast disk drive

6809 source & object	$149.00
6809 object	$ 99.00

OUTSIDE MODEM PROGRAM incl. source

UniFLEX version	$100.00
FLEX version 6800 & 6809	$ 50.00
HAYES SMART MODEM	$249.00

CROSS ASSEMBLERS for 6800,6801,6805 runs on 6809 FLEX $150.00

TV-EDIT Screen oriented Editor for FLEX 6809 $ 95.00
Please specify 5" or 8" Disk when placing order.

CHIPS

Please note when ordering IC's that a $100.00 Minimum is in effect Orders less than the minimum will be charged an additional $10.00 for handling

6116P-3	2MHZ	$13.50
2016P-2	2MHZ	$10.50
2716	1MHZ	$ 6.50
2716-1	2MHZ	$16.50
146805E2P		$25.00
68B02		$10.00
68B09		$25.00
68B21		$ 4.50
68B40		$12.00
68B50		$ 6.75
74LS640		$ 3.25

HARDWARE

NEW MEMORY PRICES TO INTRODUCE VDISK

S-R/R without memory chips	$120.00
with 8K NMOS@2MHZ	$169.00
with 16K NMOS@2MHZ	$199.00
with 32K NMOS@2MHZ	$299.00
with 48K NMOS@2MHZ	$399.00
Bare Card	$ 49.00

Extender Cards assemb. with logic aid

SS-50/50C	$ 35.00
SS-50	$ 25.00
SP-1 Prototype Board A/T	$195.00
SP-1 Bare Card	$ 49.00
SS-50 Wire Wrap Board B/C	$ 39.00
SS-50 Wire Wrap Board B/C	$ 20.00
6802 Super CPU A/T	$235.00
6802 Super CPU B/C	$ 59.00
Video RAM, B/C $49.00, A/T	$195.00
Parallel I/O, B/C $49.00, A/T	$139.00
SS-50 Backplanes 4,6,8,12,616 position @ $5.00 per slot w/o connectors	
SS-50 8 pos. BP w/o connect.	$ 39.00
Transition Card $49.00, A/T	$ 95.00
Molex Gold Male $1.60, Female	$ 1.60
Molex Tin Male $ .40, Female	$ .50

THOMAS INSTRUMENTATION

168 EIGHTH STREET — AVALON, N.J. 08202 (609) 967-4280
NJ RES. INCLUDE 5% SALES TAX
Allow 3 Weeks for Checks to Clear
CONT. USA INCLUDE $3.00 SHIPPING, CANADA $6.00, FOREIGN $12.00
MASTERCARD, VISA, and C.O.D. ACCEPTED
Send $20.00 to receive full documentation, schematics, & source listings for all boards currently in production.

'68' MICRO JOURNAL ADVERTISERS INDEX

This Index is provided as a reader service. The publisher does not assume any liability for omissions or errors.

THE CHIEFTAIN™ 5¼-INCH WINCHESTER HARD DISK COMPUTER

SO ADVANCED IN SO MANY WAYS . . . AND SO COST-EFFECTIVE . . . IT OBSOLETES MOST OTHER SYSTEMS AVAILABLE TODAY AT ANY PRICE.

● HARD DISK SYSTEM CAPACITY

The Chieftain series includes 5¼- and 8-inch Winchesters that range from 4- to 60-megabyte capacity, and higher as technology advances. All hard disk Chieftains include 64-k memory with two serial ports and DOS69D disk operating system.

● LIGHTNING ACCESS TIME

Average access time for 5¼-inch Winchesters is 70-msec, comparable to far more costly hard disk systems. That means data transfer *ten-times faster* than floppy disk systems.

● 2-MHZ OPERATION

All Chieftains operate at 2-MHz, regardless of disk storage type or operating system used. Compare this to other hard disk systems, no matter ***how*** much they cost!

● DMA DATA TRANSFER

DMA data transfer to-and-from tape and disk is provided for optimum speed. A special design technique eliminates the necessity of halting the processor to wait for data which normally transfers at a slower speed, determined by the rotational velocity of the disk.

● RUNS UNDER DOS OR OS-9

No matter which Chieftain you select . . . 5¼- or 8-inch floppy, or 5¼- or 8-inch Winchester with tape or floppy back-up . . . they ***all*** run under DOS or OS-9 with ***no need*** to modify hardware or software.

● UNBOUNDED FLEXIBILITY

You'll probably never use it, but any Chieftain hard disk system can drive up to 20 other Winchesters, and four tape drives, with a single DMA interface board!

● SMOKE SIGNAL'S HERITAGE OF EXCELLENCE

This new-generation computer is accompanied by the same ***Endurance-Certified*** quality Dealers and end-users all over the world have come to expect from Smoke Signal. And support, software selection and extremely competitive pricing are very much a part of that enviable reputation.

20-Megabyte Tape Streamer Back-Up Option

Available with all Chieftain hard disk configurations. This cartridge tape capability provides full 20-megabyte disk back-up in less than five minutes with just one command, or copy command for individual file transfers. Transfers data tape-to-disk or disk-to-tape. Floppy back-up is also available in a variety of configurations.

The Chieftain Computer Systems:

Here are the Chieftain 6809-based hard disk computers that are destined to change the data processing industry . . .

☐ **CHIEFTAIN 95W4**
4-megabyte, 5¼-inch Winchester with a 360-k floppy disk drive (pictured).

☐ **CHIEFTAIN 95XW4**
4-megabyte, 5¼-inch Winchester with a 750-k octo-density floppy disk drive.

☐ **CHIEFTAIN 98W15**
15-megabyte, 5¼-inch Winchester with a 1-megabyte 8-inch floppy disk drive.

☐ **CHIEFTAIN 9W15T20**
15-megabyte, 5¼-inch Winchester with a 20-megabyte tape streamer.

Write or call today for details (including the remarkably low prices) on the total Chieftain Series . . . and on dealership opportunities.

SMOKE SIGNAL BROADCASTING®
31336 VIA COLINAS
WESTLAKE VILLAGE, CA 91362
TEL. (213) 889-9340

Name _____

Company _____

Address _____

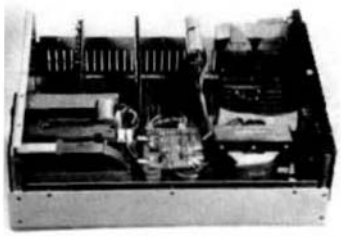
City _____ State _____ Zip _____

Telephone ( ) _____

FLEX - OS-9 LEVEL ONE - UNIFLEX - OS-9 LEVEL TWO

ONLY GIMIX Systems can be configured to run any of these.

GIMIX systems utilize the most powerful 6809 operating systems: FLEX, UniFLEX, OS-9 LEVEL ONE and TWO -- the systems the PROs use. This means a wide selection of software to choose from as well the ability to develop sophisticated, multi-user/multi-tasking programs on your GIMIX System.



The GIMIX CLASSY CHASSIS™ consists of a heavy-weight aluminum mainframe cabinet which provides more than ample protection for the electronics and 1 or 2 optional 5¼" drives.

Backpanel connectors can be added for convenient connection of terminals, printers, drives and other peripherals.

A 3 position locking keyswitch enables users to disable the front panel reset button to prevent accidental or unauthorized tampering with the system.

The GIMIX system mother board provides fifteen 50 pin slots and eight 30 pin I/O slots -- the most room for expansion of any S50 system available. The on board baud rate generator features 11 standard baud rates, 75 to 38.4K, for maximum versatility and compatibility with other systems. Extended address decoding allows the I/O block to be addressed anywhere in the 1 megabyte address space. All components feature Gold plated connectors for a lifetime of solid connections. All boards are fully buffered for maximum system expansion.

Each GIMIX Mainframe System is equipped with an industrial quality power supply featuring a ferro-resonant constant voltage transformer to insure against problems caused by adverse power input conditions such as A.C. line voltage fluctuations etc. The supply provides 8 volts at 30 amps and plus or minus 16 volts at 5 amps, more than enough capacity to power a fully loaded system and two internal drives.

The 2MHz GIMIX 6809 PLUS CPU board includes a time of day clock with battery back-up and 6840 programmable timer to provide the programmer with convenient, accurate time reference. Later addition of 9511 or 9512 arithmetic processors is provided for on the board. The unique GIMIX design enables software selection of either OS-9 or FLEX, both included in many complete GIMIX systems.

GIMIX STATIC RAM boards require no complicated refresh timing cycles or clocks for data retention. GIMIX memory boards are guaranteed for 2 MHz operation with no wait state or clock stretching required.

Our low power NMOS RAM requires less than 3/4 amp at 8V for a fully populated 64K board. For critical situations, our non-volatile 64K byte CMOS static RAM boards with built in battery back-up retain data even with system power removed. A fully charged battery will power this board for a minimum of 21 days. A write protect switch permits CMOS boards to be used for PROM/ROM emulation and software debugging.

The GIMIX DMA controller leaves the processor free to perform other tasks during disk transfers - an important feature for multi-user/multi-tasking systems where processor time allocation is critical. The DMA board will accomodate up to 4 drives 5¼" or 8" in any combination running single or double density single or double headed. Programmed I/O Disk Controllers are also available.

GIMIX systems are designed with ultimate RELIABILITY in mind. You can choose from the below featured systems or select from our wide variety of components to build a custom package to suit your needs.

GIMIX 2MHz 6809 System including: CLASSY CHASSIS, 6809 PLUS CPU BOARD, 56KB STATIC RAM, 2 SERIAL PORTS W/CABLES, GMXBUG MONITOR, FLEX, and OS-9 LEVEL 1 ..... **$3248.49**
FOR TWO 5¼" 40 TRACK DSDD DRIVES ADD ..... **$ 900.00**
GIMIX 128KB WINCHESTER SYSTEM including: CLASSY CHASSIS, 6809 PLUS CPU BOARD, 128KB STATIC RAM, 4 SERIAL PORTS W/CABLES, 5¼" 80 TRACK DSDD FLOPPY DISK DRIVE, 19 MB 5¼" WINCHESTER HARD DISK, OS9 LEVEL2, EDITOR AND ASSEMBLER **$8998.09**

50HZ Versions Available. 8" Drives Available — Contact GIMIX for Prices and Information.

The Sun Never Sets On A GIMIX!

GIMIX users are found on every continent, including Antarctica. A representative group of GIMIX users includes: **Government Research and Scientific Organizations** in Australia, Canada, U.K. and in the U.S.; NASA, Oak Ridge, White Plains, Fermilab, Argonne, Scripps, Sloan Kettering, Los Alamos National Labs, AURA. **Universities:** Carleton, Waterloo, Royal Military College, in Canada; Trier in Germany; and in the U.S.; Stanford, SUNY, Harvard, UCSD, Mississippi, Georgia Tech. **Industrial users** in Hong Kong, Malaysia, South Africa, Germany, Sweden, and in the U.S.: GTE, Becton Dickinson, American Hoechst, Monsanto, Allied, Honeywell, Perkin Elmer, Johnson Controls, Associated Press, Aydin, Newkirk Electric, Revere Sugar, HI-G/AMS Controls, Chevron. **Computer mainframe and peripheral manufacturers,** IBM, OKI, Computer Peripherals Inc., Qume, Floating Point Systems. **Software houses;** Microware, T.S.C., Lucidata, Norpak, Talbot, Stylo Systems, AAA, HHH, Frank Hogg Labs, Epstein Associates, Softwest, Dynasoft, Research Resources U.K., Microworks, Meta Lab, Computerized Business Systems.

1337 WEST 37th PLACE
CHICAGO, ILLINOIS 60609
(312) 927-5510
TWX 910-221-4055

GIMIX inc.

The Company that delivers

Quality Electronic products since 1975.

FLEX & RS COLOR COMPUTER

If you are tired of playing games on your RS-80C™ Color Computer, or find that you are handicapped by the limitations of the RS BASIC in trying to write a Program that will allow you to actually USE the Color Computer as a COMPUTER, YOU ARE READY TO MOVE UP TO THE FLEX9™ Operating System. If you want to have REAL PROGRAMMING POWER, using an Extremely Powerful Business BASIC, PASCALs, C Compilers, a full-blown Macro Assembler with a Library capability so you are not continuously "reinventing the wheel", YOU ARE READY TO MOVE UP TO THE FLEX9™ Operating System. If you would like to see if YOU REALLY COULD USE A COMPUTER IN YOUR BUSINESS, or begin to make your Computer start PAYING IT'S OWN WAY by doing some Computer Work for the millions of small busines es around you, such as Wordprocessing, Payroll, Accounting, Inventory, etc., then YOU ARE READY TO MOVE UP TO THE FLEX9™ Operating System. How?? DATA-COMP has the way!

DATA-COMP's FLEX9™ Conversion for the TRS-80C™ Color Computer was designed for the SERIOUS COMPUTER USER; with features like greatly increased Display Screens, WITH Lower Case Letter , so you can put a FULL Menu on ONE Screen, or see SEVERAL Paragraphs at the same time; with features like providing a FULL Keyboard so you have FULL Control of your Computer AND it's Programs NATURALLY, without needing a chart to see what Key Combination will give you what function; with USER ORIENTED functions to make using the Operating System natural, like having the Computer AUTOMATICALLY determine what type of Disk is being used in what type of Disk Drive and working accordingly, rather that you have to specify each and every thing for it, or like having the Computer work with the Printer you ha e been using all alo g without you having to tell the new Operating System what is there; etc., etc., etc.

DATA-COMP has everything you need to make your TRS-80C™ Color Computer WORK for YOU; from Parts and Pieces to Full, Ready To Use SYSTEMS. DATA-COMP designs, sells, services, and SUPPORTS Computer SYSTEMS, not just Software. CALL DATA-COMP TODAY to make your Computer WORK FOR YOU!

System Requirements

FLEX9 Special General Version x Editor & Assembler (which normally sell for $50.00 ea.)	$150.00
F-MATE(RS) FLEX9 Conversion Rout. for the RS Disk Controller when purchased with Special General FLEX9 Sys.	$69.95
when purchased without the General FLEX9 Sys.	$79.95
Set of Eight 64K RAM Chips w/Mod. Instructions	$99.95
Color Computer with 64K RAM and EXT. BASIC	$549.95
Color Computer with 16K RAM	$375.95
Color Computer with 16K RAM and EXT. BASIC	$465.95

Now Available
Enhanced F-MATE Version 2.1

SPECIAL SYSTEM PACKAGES

64K Radio Shack COLOR COMPUTER, Radio Shack COLOR DISK CONTROLLER, a Disk Drive System, Special General Version of FLEX9™, F-MATE(RS)™ and a Box of 10 Double Density Diskettes; a COMPLETE, ready to run SYSTEM on your Color TV Set. $1249.95

DISK DRIVE PACKAGES, etc.

These Packages Include the Radio Shack Disk Controller, Disk Drives with Power Supply and Cabinet, and Disk Drive Cable:

PAK #1 — 1 Single Sided, Double Density Sy .	$499.95
PAK #2 — 2 Single Sided, Double Density Sys.	$769.95
PAK #3 — 1 Double Sided, Double Density Sys.	$599.95
PAK #4 — 2 Double Sided, Double Density Sys.	$949.95

PARTS AND PIECES

Radio Shack Disk Co troller	$179.95
1 ea. Single Sided, Double Density Disk Drive	$249.95
1 ea. Double Sided, Double Density Disk Drive	$349.95
Single Drive Cabinet with Power Supply	$89.95
Double Drive Cabinet with Power Supply	$109.95
Single Drive Disk Cable for RS Controller	$24.95
Double Drive Disk Cable for RS Co troller	$34.95
Micro Tech. Prods., Inc. **LOWER CASE ROM** Adapter	$74.95
Radio Shack BASIC Version 1.1 ROM	$34.95

SOFTWARE

CHESS 6809

Requires FLEX™ and one of the following CRT terminals

Now Runs On Any Type Terminal

Features:

- **Two display boards.**
- **Four levels of play.**
- **Point scoring system.**
- **Play white or black.**
- **Change or set-up boards/piece positions.**
- **Forfeit move.**
- **Swap sides.**
- **Make move and swap sides.**
- **Change skill level.**
- **Stop and restart game.**
- **Solve 'Mate in 1-2-3-4' moves.**

$79.95 Specify 5" or 8" disk

This is one of the strongest CHESS programs running on any microcomputer, estimated USCF Rating 1600 + .

Note: Personal checks allow 3-4 week delivery.

DIET-TRAC Forecaster

A Diet Planning and Analysis Program

DIET-TRAC Forecaster is a program that plans a diet in terms of either calories and percentage of carbohydrates, proteins and fats (C.P.F. %) or grams of Carbohydrate, Protein and Fat food exchanges of each of the six basic food groups (vegetable, bread, meat, skim milk, fruit and fat) for a specific individual.

Sex, Age, Height, Present Weight, Frame Size, Activity Level and Basal Metabolic Rate for normal individuals are taken into account. Ideal weight and sustaining calories for any weight of the above individual are calculated. When a weight goal is given (either gain or loss), and a calorie plan is agreed upon between the computer and the individual, the number of days to reach the weight goal is projected. The starting and ending rate of weight loss is calculated, and a daily calendar with each day's predicted weight for a 30-day period is printed.

FLEX VERSION	**$59.95**
UniFLEX VERSION	**$89.95**

PRINTERS

The Epson MX-80
$495.00

The Epson MX-100
$725.00

MX-70 $355.00 MX-80 FT $575.00

☆

Color Computer External Terminal Program

$19.95

☆

Color Computer Stylograph Running Under F-Mate

$195.00

MasterCard VISA

DATA-COMP SOUTH EAST MEDIA

P.O. Box 794 HIXSON, TN 37343

1-615-842-4601

Verbatim Diskettes.

5" Soft Sector Disks

Single Side Single Density	$2.75 ea.
Single Side Double Density	$2.75 ea.
Double Side Double Density	$4.92 ea.
Plastic Storage Box	$2.00 ea.

8" Soft Sector Disks

Single Side Single Density	$3.75 ea.
Single Side Double Density	$4.10 ea.
Double Side Double Density	$4.75 ea.
Plastic Library Box	$5.00 ea.

Foreign Orders Add 10% Surface—20% Air Mail

WINCHESTER BACKUP UTILITIES

The following utilities allow the backup of any size disk system to any size diskettes.

By simply inserting diskettes as requested by COPYMULT, a large disk system (Winchester, etc.) may be downloaded to your present floppy disk system, any size. No need to fiddle with directory deletions or any of the other tedious operations that must be done using a normal copy routine.

COPYMULT.CMD understands normal "copy" syntax and always keeps up with files already copied by maintaining directories for both host and receiving disk system, thus eliminating hours of tedious keyboard entries and other time consuming cleanup chores.

BACKUP.CMD is a special program that downloads "random" type files, any size.

RESTORE.CMD a special program to restructure copied "random" files for copying, or recopying back to the host system.

FREELINK.CMD a "bonus" utility that "relinks" the free chain of a floppy or hard disk thereby eliminating fragmentation.

**** Completely documented source files included.**

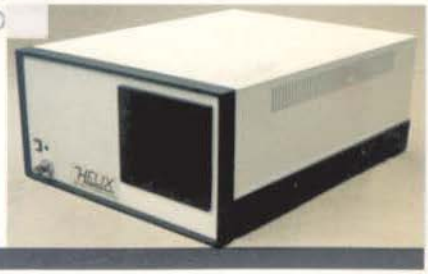
**** ALL 3 Programs $99.50 on 8" diskette**

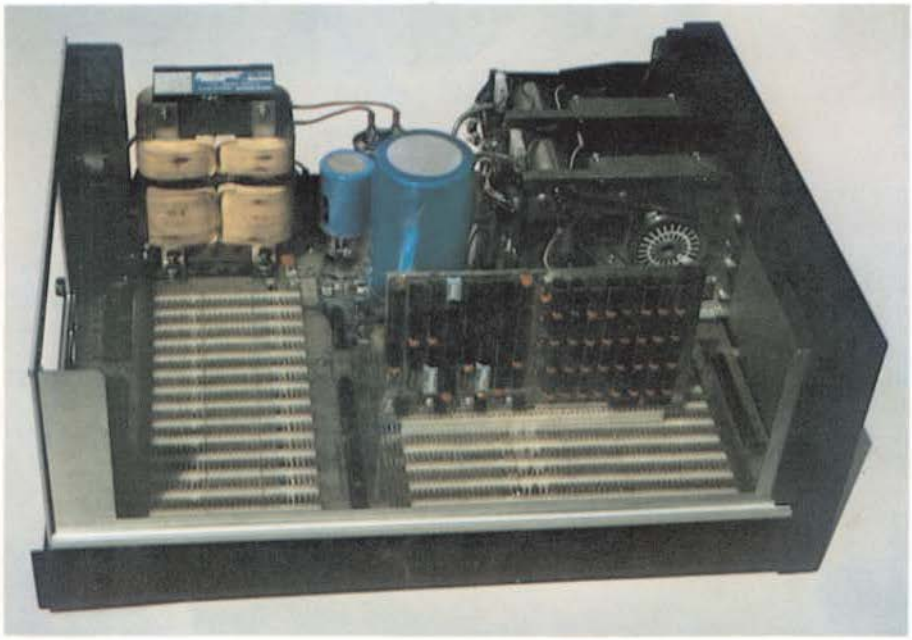
Data-Comp—South East Media & 68 Micro Journal Are Divisions of CPI

68 Micro Journal
5900 Cassandra Smith Rd.
Hixson, TN 37343

Second Class Postage Paid
At Chattanooga, TN
ISSN 0194-5025

HELIX™





THE MAINFRAME

- Industry Standard Optima™ Cabinet
- Largest Constant Voltage Power Supply in the industry
- S-64 Bus gives 16 Bit Power and S-50 Bus Compatibility
- 10 Main (S-64) Slots
- 14 I/O (S-30) Slots plus 2 On-board
- On-board Baud Rate Generator to 38.4Kb
- Space and Power for two 5¼" Disk Drives
- Full Address Decoding for I/O Slots
- Two RS-232 Serial and Two parallel Ports On-board
- Single Board Construction for Reliability
- Faraday Shielded Bus Lines give "Text Book Clean" Signals

THE PROCESSORS

6809

- Standard 2 MHz Operation
- Standard DAT Compatible with GIMIX and SWTPC
- Standard 6840 Interval Timer
- Standard 1K Scratchpad RAM
- Standard Clock/Calendar with Battery
- Provision for Programmers Console

68000

- Standard 8 MHz Operation
- Memory Management Hardware
- Provision for Programmers Console
- 16 Bit Power and 8 Bit Compatibility

The HELIX™ computer system represents the latest advance in S-50 bus computer systems. Relying on the physical nature of S-50 bus connectors to guarantee compatibility, the HELIX adds 14 bus lines (becoming S-64) to allow a 68000 processor to operate with full 16 bit data transfer and 24 bit addressing, while at the same time providing full interchangability with existing S-50 components.

Offered with a selection of processors, memories, and peripheral controllers, a HELIX system can be configured for applications ranging from advanced hobbyist to multiterminal time-sharing.

Designed to offer the utmost in speed, reliability, and utility at a reasonable price, it represents a new standard of quality for those who require a professionally designed computer for professional use.

THE POWER SUPPLY

- Ferro-resonant Transformer for Line Noise and Under-Voltage Protection
- Conservative 25 Amps at 8.5 Volts
- Conservative 5 Amps at ±16 Volts
- Conservative Component Rating for Reliability

THE COMPONENTS

- Fully Socketed
- Gold Plated Bus Connectors
- Only "B" Series 68XX Components Used
- Only Top Grade Logic Circuits Used
- Industrial Grade Components Throughout

THE MEMORIES

DM-64

- Field Proven
- Proprietary Memory Control Logic
- Fully Transparent Refresh
- Tested at 2.5 MHz Operation

DM-512

- 512K Bytes on a Single S-64 Board
- 16 Bit Power and 8 Bit Compatibility
- Runs in Existing S-50 Systems where Physical Space Allows
- Full 24 Bit Addressing
- Fully Transparent Refresh

THE PRICES

Because of the variety of configurations possible, full pricing cannot be given. Representative prices are:

- 64K 6809 HELIX $1995
- 64K 68000 HELIX $2595
- 512K 6809 HELIX $4450
- 512K 68000 HELIX $4995

HAZELWOOD COMPUTER SYSTEMS

907 E. Terra, O'Fallon, Missouri 63366 (314) 281-1055

Dealer and OEM Inquiries Invited. We support our Dealers.

Optima is a Trademark of Scientific-Atlanta